MRP EXPLOSION FOR THE IBM COMPUTER

by

SUNSERN CHUANPRAPUN

B.S., King Mongkut's Institute of Technology, 1981

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

Approved by:

Major Professor

LD
2668
.T4
IE
1987
C58
c. 2

A11207 307608

TABLE OF CONTENTS

page

LIST OF FIGURES iii

ACKNOWLEDGMENTS v

INTRODUCTION 1

THE OBJECTIVES OF THIS THESIS 2

MRP DEFINITION 3

MRP HISTORY 5

LANGUAGE AND COMPUTER USED IN THE STUDY 8

SETTING UP RAMDISK 8

SAMPLE PROBLEM 9

DATA BASE METHOD 13

PART LIST FILE 15

NUMERICAL EXAMPLE FOR PART LIST RECORD FILE 22

BILL OF MATERIAL RELATIONSHIPS FILE 25

NUMERICAL EXAMPLE FOR BOM RELATIONSHIPS DATA BASE FILE 31

MASTER PRODUCTION SCHEDULE FILE 34

NUMERICAL EXAMPLE FOR MASTER PRODUCTION SCHEDULE FILE 34

LEVEL CODE ASSIGNMENT PROCEDURE 40

NUMERICAL EXAMPLE FOR ASSIGNED LEVEL CODE PROCEDURE 42

MRP EXPLOSION PROCEDURE 45

NUMERICAL EXAMPLE FOR MRP EXPLOSION PROCEDURE ... 48

PROGRAM LIMITATIONS 55

SPEED OF THE MRP PROGRAM 55

CONCLUSION 58

APPENDIX A - PROGRAMS (DEBUGGING) 59

APPENDIX B - THE PROGRAM DESIGN LANGUAGE OR PSEUDOCODE 116

REFERENCES 163

LIST OF FIGURES

		Page
Figure 1.	Part List number and quantities shown in form of Production Tree	10
Figure 2.	The Stock Status	11
Figure 3.	Independent Demand Requirements	12
Figure 4.	Part List procedure block diagram	16
Figure 5.	Material Requirements Planning database screen menu	17
Figure 6.	Inventory Records File edit screen	18
Figure 7.	Procedure to add a record into the Part List data and index files	20
Figure 8.	Procedure to delete a record from the Part List data and index files	21
Figure 9.	An example for entering part number A into the Inventory Records File	23
Figure 10.	An example for entering part number F into the Inventory Records File	24
Figure 11.	The BOM Relationship procedure block diagram	26
Figure 12.	BOM Relationship File edit screen	27
Figure 13.	Procedure To add BOM Relationship Records	29
Figure 14.	Procedure To Delete BOM Relationship Records	30
Figure 15.	An example for entering Part Make A and Part Needed C into the BOM Relationship File	32
Figure 16.	An example for entering Part Make A and Part Needed B into the BOM Relationship File	33
Figure 17.	The Master Production Schedule procedure block diagram	35
Figure 18.	Procedure To Add Information to the Master Production Schedule Records	36

Figure 19. Procedure To Delete Master Production Schedule Records 37

Figure 20. An example for entering Part A into the Master Production Schedule File ... 38

Figure 21. An example for entering Part F into the Master Production Schedule File ... 39

Figure 22. An example for entering Part C into Master Production Schedule File 40

Figure 23. The BOM Assigned Level Code procedure block diagram 41

Figure 24. An example of the assigned level code procedure 43

Figure 25. The computer output from procedure BOM assigned level code 44

Figure 26. The MRP explosion procedure block diagram 47

Figure 27. The MRP result of part number A 49

Figure 28. The MRP result of part number F 50

Figure 29. The MRP result of part number B 51

Figure 30. The MRP result of part number D 52

Figure 31. The MRP result of part number C 53

Figure 32. The MRP result of part number E 54

Figure 33. Testing the speed of the MRP program at procedure BOM assigned level code (performance time used to finish the procedure) 57

Figure 34. Testing the speed of the MRP program at procedure MRP explosion (performance time used to finish the calculation result of one part number) 57

ACKNOWLEDGMENTS

I would like to thank Dr. L. E. Grosh, who was by major advisor, for continuous guidance and invaluable advice in planning the study and preparing this thesis. I would also like to thank Dr. C. L. Hwang and Dr. Fredric C. Appl for helping and serving on my advisory committee.

Finally, I want to say thanks to my parents. Additional thanks to Dr. Frank A. Tillman head Department of Industrial Engineering, Dr. R. Basette and Mrs. Basette, my brother, and Somjit Rerkrai, for their continuous encouragement.

INTRODUCTION

Material Requirements Planning (MRP) or Time-phased Material Requirements Planning has been known for many years. MRP is an important technique which is useful for determining needs for dependent demand items (such as raw materials, components, purchased parts, etc.), planning inventory levels, and scheduling production. The objective of MRP is to minimize on-hand physical stocks of raw materials and component parts while providing assurance of the availability of needed stocks for planned production runs with a high degree of confidence as pointed out by R.J. Compbell and T.M. Porcano.

MRP techniques have been adopted by thousands of companies and are being taught and researched in most major universities not only in the United States but also in many countries around the world. Since the 1960s, MRP software programs have been created to run with a large computer.[1] Today, the technology of computers has been changed from large computers to small computers. Many companies and researchers are developing MRP programs in order to use with small computers. So several different MRP software packages have recently appeared on the market for use with micro and minicomputers. These tools and techniques offer business managers the opportunity to control their business efficiently. However, MRP software packages on the market require large computer.

THE OBJECTIVES OF THIS THESIS

The objectives of this thesis are

1 To write Turbo Pascal programs to perfrom the basic MRP explosion.

2 To design and develop the data base files, and their associated input and output formats.

3 To compare the speed of MRP programs with different disk drives.

MRP DEFINITION

Material Requirements Planning (MRP) is a techniques or computerized data information system that can support scheduling and manufacturing functions. Millard has given the definition of MRP, MRP is a series of computer programs that essentially matches material requirements with inventory levels and existing replenishment purchase orders and work orders over a time horizon, determines where discrepancies exist, and makes recommendations to restore a balance between the requirements in a process called an "explosion" of the master production schedule. The computer programs first determine gross needs for each part and then subtract inventories and/or firm replenishment orders, if any, to compute net requirements. The explosion and netting process begins with the top assembly items and proceeds down the assembly structure and ultimately ends with raw materials so that net requirements will have been determined for every part number. Orlicky explained "Material Requirements Planning (time phasing is implied by the term) evolved from an approach to inventory management in which the following two principles are combined:

1. Calculation (versus forecast) of component item demand.
2. Time phasing, i.e., segmenting inventory status data by time.

The term component item in MRP covers all inventory items other than products or end items. Requirements for end items are stated in the master production schedule, and they are derived from forecasts, customer orders, field warehouse requirements, interplant orders, etc. Requirements for all component items (including raw material), and their timing, are derived from this schedule by the MRP system."[2]

MRP is normally applied to a computer-based system because of the large amount of data handling that MRP entails.[3] The basic inputs to an MRP system are the master production schedule, bill of materials file, and inventory-records file[4] and "the principle outputs are specifically the reschedule of planned (future) order releases, order action recommendations calling for release of current orders, and cancellation or reschedule of open (released) orders." as pointed out by APICS training aid.[5]

MRP HISTORY

The basic logic of Material Requirements Planning began in the 1920s continued through the 1940s as observed by Plossl.[6] As products became more and more complex with more components, manual calculations to perform MRP became more difficult and more time consuming. At this time Orlicky and Plossl related that another technique had been developing called "the cult of the order point".[7] The order point technique was used in the 1940s and persisted into the 1950s up to the mid 1960s.[8]

Orlicky, Plossl, and Wight point out that the problems associated with order point system were excess inventory, frequent shortages, and high costs.[9] In the mid 1960s, the first mainframe computer became available, allowing manufacturing inventory management applications to be developed.[10] Plossl related that "A few pioneers in the 1950s and early 1960s had tried to lead the way with the computer then available; outstanding among there were Joe Orlicky and Jack Chobaniam at J.I. Case company in Racine, Wisconsin and some aerospace and automotive companies made a half-harted effort, using gross requirements instead of net. In the late 1950s".[11] Gilbert and Schonberger observed that "The first batch MRP system was developed at American Bosch Company of Springfield, Massachusetts. This system was actually running on a bi-weekly basis in 1959."[12] Plossl also tried to apply the MRP technique using an IBM Ramac 305

at Gilbert and Barker but was hampered by the lack of adequate bills of material.[13]

Plossl pointed out "The earliest literature on MRP was produced by the computer manufacturers, mainly IBM and UNIVAC. Its principal purpose was to sell computers".[14] By this time the MRP software ran well but had little useful information on solving the problems of applying the technique.[15]

Many researchers tried to develop new techniques to improve the MRP programs. In 1977, Orlicky and McDonald explained about regenerative material requirements planning system, "the design of any net change MRP system makes it necessary that significantly more data be stored and, overtime, repeatedly accessed by the system. How these data are organized and managed is therefore critically important for efficient retrieval, processing, and maintenance."[16] By this time IBM had created a data base management technique called Data Language/I or DL/I.[17] As pointed out by Orlicky and McDonald, for purposes of the MRP application, DL/I is the successer to the original IBM Bill of Material Processer and the subsequent Data Base Organization and Maintenance Program.[18] "The guiding principle of DL/I "architecture" is that records that are used together are stored in adjacent areas so as to minimize total retrieval time. The data required by the system would therefore not be stored in separate files, and the data would be organized in such a way as to minimize the number of accesses required for the

retrieval of related data."[19] as pointed out by Orlicky and McDonald.

In 1983 at Kansas State University, an MRP program was created to use with a microcomputer by Karen May and L.E. Grosh. The program was written in Waterloo Fortran and used the Commodore Supper Pet computer model 8050. The purpose of creating this program was to investigate the possibility of a microcomputer being capable of handing the MRP program.[20] This procedure was too slow to be of practical classroom use.

LANGUAGE AND COMPUTER USED IN THE STUDY

All of the programs used in this study were written in Turbo Pascal and developed by using a IBM PC or IBM PC compatible which has the 8088 or 8086 CPU. The operating system used was the PC-DOS or MS-DOS version 2.XX or higher. The Vdisk.com from PC-DOS or MS-DOS was used to create RAMdisk. The memory requirement to create and run the program was at least 640KB of RAM. The source programs were written using Edit mode in the Turbo Pascal Compiler software version 2.00 and also complied with Turbo Pascal Compiler software version 2.00. The data base subprograms in Turbo Tool Box software version 2.00 by Bordland International Inc. were used to develop data bases in our program. The Turbo Pascal Compiler software version 3.0 does not access Turbo Tool Box version 2.0 correctly. The version of the Turbo Pascal language and the Turbo Tool Box data base subprograms are described in Turbo Pascal Reference Manual version 2.00 and in Turbo Tool Box Reference Manual version 2.00.

SETTING UP RAMDISK

The purpose of creating RAMdisk is to increase the program speed. RAMdisk was set up by adding command "DEVICE=VDISK.SYS D: 384 256 64" in the CONFIG.SYS file. This procedure will create a RAMdisk size of 384KB at disk drive D:. This RAMdisk is used to hold data files while the

program runs, and 256KB will be used for MRP program execution.

SAMPLE PROBLEM

The sample problem used in this study was taken from an APICS training aid in the MRP work exercise, the sample problem is presented in the following purposes:

1. To explain the different data files;
2. To show how to assigned level codes;
3. To explain the different steps to perform the MRP explosion;
4. To explain how the MRP software works.

The sample problem data are shown in Figures 1, 2, and 3.

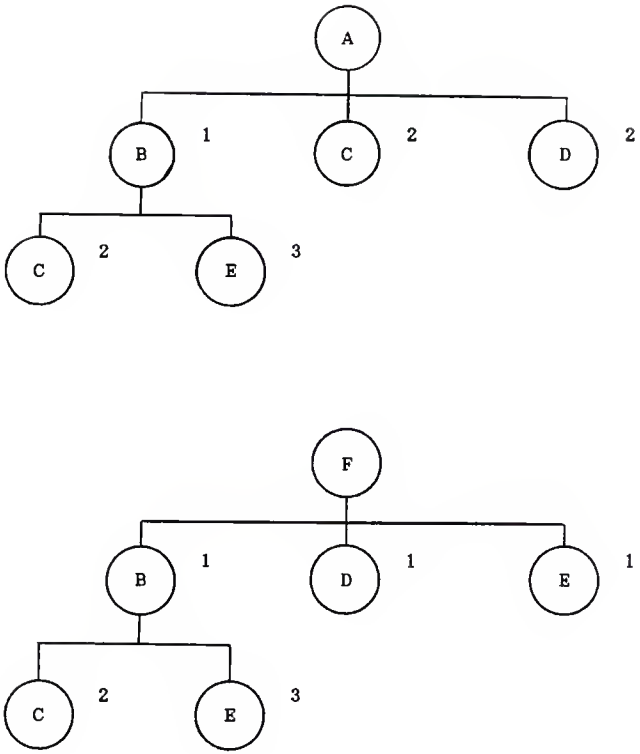

Figure 1 Part List number and quantities shown in form of Production Tree

ITEM NO.	LOW LEVEL CODE	LEAD TIME (WEEKS)	ON HAND	ON ORDER	SAFETY STOCK	ALLOCATED	ORDER POLICY
A	1	2	0	0	0	0	LOT FOR LOT
F	1	1	0	0	0	0	LOT FOR LOT
B	2	1	100	0	0	0	LOT FOR LOT
D	2	1	150	100 DUE IN 1ST WEEK	0	100	160
C	3	2	120	0	15	0	150
E	3	1	120	0	0	0	140

Figure 2 The Stock Status

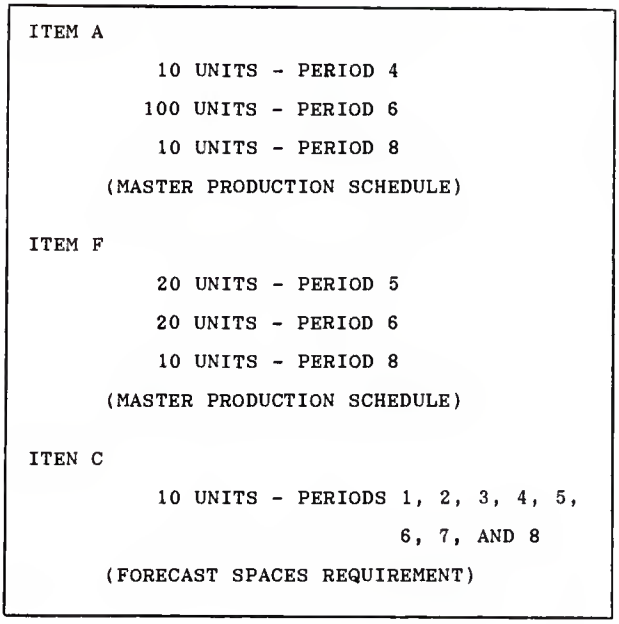
ITEM A
10 UNITS - PERIOD 4
100 UNITS - PERIOD 6
10 UNITS - PERIOD 8
(MASTER PRODUCTION SCHEDULE)
ITEM F
20 UNITS - PERIOD 5
20 UNITS - PERIOD 6
10 UNITS - PERIOD 8
(MASTER PRODUCTION SCHEDULE)
ITEN C
10 UNITS - PERIODS 1, 2, 3, 4, 5,
6, 7, AND 8
(FORECAST SPACES REQUIREMENT)

Figure 3 Independent Demand Requirements

DATA BASE METHOD

B-Tree techniques from the Turbo Tool Box have been used to develop all of the data base files used in the MRP explosion process (for more details of the B-Tree technique see Turbo Tool Box Reference Manual version 2.00). The B-tree technique is one of the several data base techniques that can be used with an MRP program, the advantage of the B-Tree technique is the fast speed for searching records.

Our basic MRP data base system design is as follows:

A. Part List File

* Part Number
* Lot Size
* Lead Time
* On Hand
* On Order
* Safety Stock
* Allocated

B. Bill-of-Material File

* Parent part made
* The sub-parts required to make one parent part
* The quantity of sub-parts per parent part

C. Master Production Schedule File

* Part number

* Independent Demand Requirements for each period

D. Results of MRP File

* Part number
* Low level code
* Lot size policy
* Lead time
* On hand
* On order
* Safety stock
* Allocated
* Gross requirements
* Schedule receipts
* Projected on hand
* Net requirements
* Planned order receipts
* Planned order releases

PART LIST FILE

The Part List file or Inventory Records file is used to keep the list of part numbers and the information that is associated with part number such as lot size, lead time, on hand etc. In order to create and organize the Part List file, the program called "DATA1.PL" consisting of six procedures (procedures Database, Makescreen, Update, Dec, Addp, and Delete) needs to be used. See Figure 4 for the Part List file block diagram. Start at procedure Database, this procedure is used to create MRP database menu screen option see Figure 5 for MRP database screen on CRT, see APPENDIX A-10 and APPENDIX B-10 for program debugging and the procedure Database' pseudo code. When the update Part list option is selected the procedure calls procedure Makescreen for making a frame on the CRT, see APPENDIX A-8 and B-8, and then procedure Update creates information needed for each record (information associated with part number) and also creates options such as add record, delete record, etc. see Figure 6 for Part list file format screen, see APPENDIX A-2 and B-2. From here if the add record option has been selected the procedure Addp will create a new record. The procedure Addp uses part number as a key index when a new record has been entered, first the procedure Addp needs to check if the new part number is already assigned or not, if the new part number is not assigned, the procedure will add a new data record into the Part list data file (information associated with part number

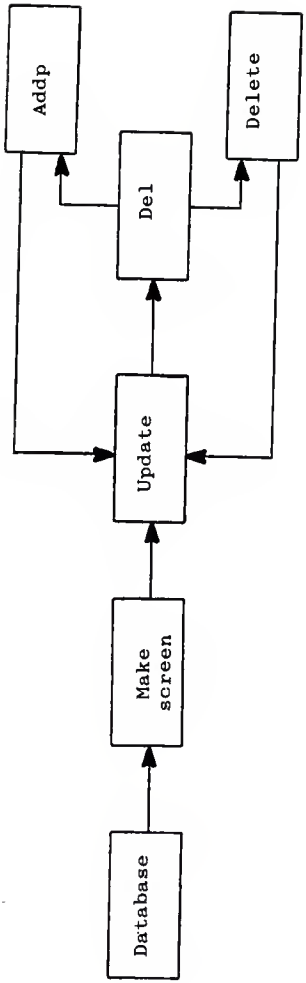

Figure 4 Part List procedure block diagram

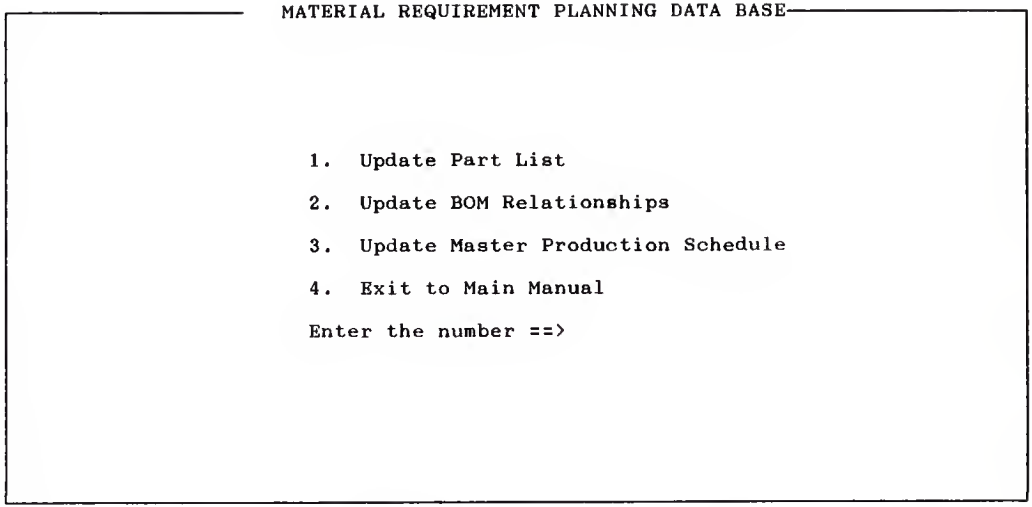
MATERIAL REQUIREMENT PLANNING DATA BASE
1. Update Part List
2. Update BOM Relationships
3. Update Master Production Schedule
4. Exit to Main Manual
Enter the number ==>

Figure 5 Material Requirements Planning database screen menu

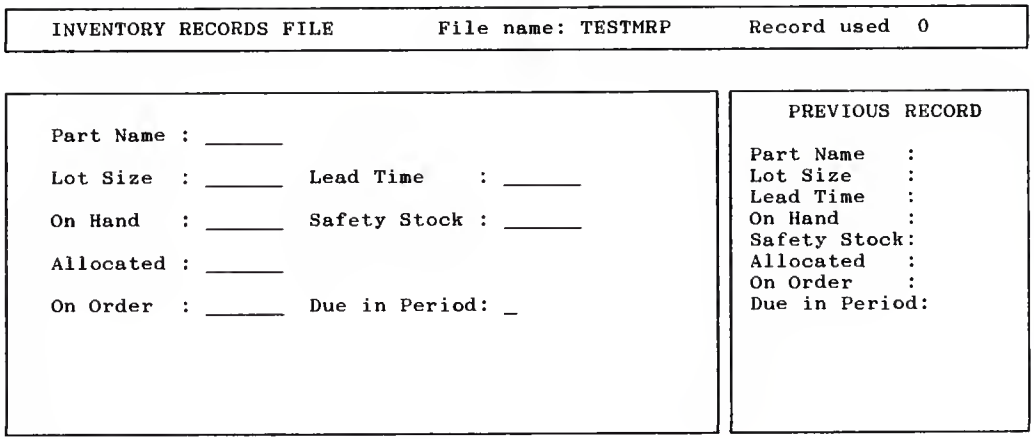

Figure 6 Inventory Records File edit screen

will be kept in this file) and the Part List index file (part number as a key index will be kept in this file). If the delete record option has been selected the procedure Delete will be performed (see APPENDIX A-2 and B-2), the procedure Delete uses part number as a key index for finding the record that we want to delete. When the part number to be deleted has been entered the procedure will find that part number and if the part number is in the index file, the procedure Delete will delete that part number in the Part List index file and then delete information associated with part number in the Part List data file, see APPENDIX A-2 and B-2. After each add or delete has been performed the program will loop back to procedure Update, for the next add or delete record, or return to the main program. The program design language is called "pseudo code", which was introduced by Pressman in the Software Engineering[21], for the procedure "add" and "delete" in "Part List file" the pseudo code is given in Figures 7 and 8.

PROCEDURE ADD PART LIST

This procedure is used to add new records to the part list file. The information to be added is part number, lot size, lead time, on hand, safety stock, allocated, on order, and period. The procedure uses part number as a key to search in the part list index file, if part number is not found, the procedure will add the record into part list data file and part number into part list index file.

```
DECLARE  part number, answer AS STRING
         status, lot size, lead time, on hand,
         safety stock, allocated, on order AS INTEGER
         part list file AS DATA FILE
         part index file AS INDEX FILE

BEGIN
  open part list file
  open part index file
  REPEAT UNTIL answer = no
    enter part number to add
    IF part number is in the part list index file
      THEN BEGIN
        set status = 0
        enter lot size, lead time, on hand,
              safety stock, allocated and on order
        add data record to part list file
        add part number to part list index file
          END
      ELSE
        print part number already assigned
    ENDIF
  ENDREPEAT
  close part list file
  close part index file
END
```

Figure 7 Procedure to add a record into the Part List data and index files

```
PROCEDURE DELETE PART LIST

     This procedure is used to delete records in the
     part list file. The procedure uses part number as a
     key to search in the part list index file. If part
     number is found, the procedure will delete both the
     part list data file record and part list index file
     record.


     DECLARE  part number, answer AS STRING

     BEGIN
       open part list file
       open part index file
       REPEAT UNTIL answer = no
         enter part number to delete
         IF part number has been found in the part list
            index file
           THEN BEGIN
             delete part number in the part list index file
             delete data record associated with part number
             in the part data file
                END
           ELSE
             print part number not found in the part index
             file
         ENDIF
       ENDREPEAT
       close part list file
       close part index file
   END
```

Figure 8 Procedure to delete a record from the Part List data and index files

NUMERICAL EXAMPLE FOR PART LIST RECORD FILE

To illustrate how to enter the data into Part List records file. The data from Figure 2 will considered. Part Number, Lead Time, On Hand, On Order, Safety Stock, Allocated, and Order Policy needed to be prepared for data entries. Call the program MRP and follow the menu instructions. Figures 9 and 10 are shown the example of entering Part Name "A" and "F" into the Inventory Record file (Part List record file).

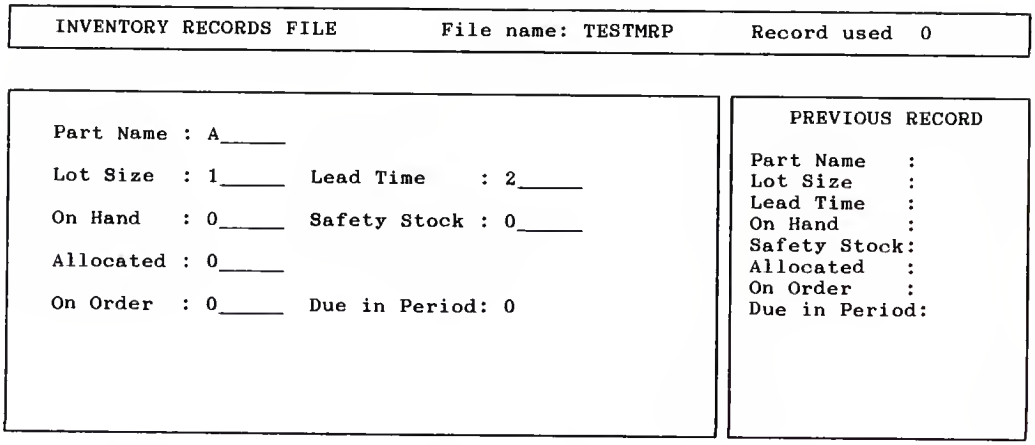
INVENTORY RECORDS FILE File name: TESTMRP Record used 0
Part Name : A
Lot Size : 1 Lead Time : 2
On Hand : 0 Safety Stock : 0
Allocated : 0
On Order : 0 Due in Period: 0
PREVIOUS RECORD
Part Name :
Lot Size :
Lead Time :
On Hand :
Safety Stock:
Allocated :
On Order :
Due in Period:
Command ? : A)dd D)elete Q)uit:

Figure 9 An example of entering part number A into the Inventory Records File

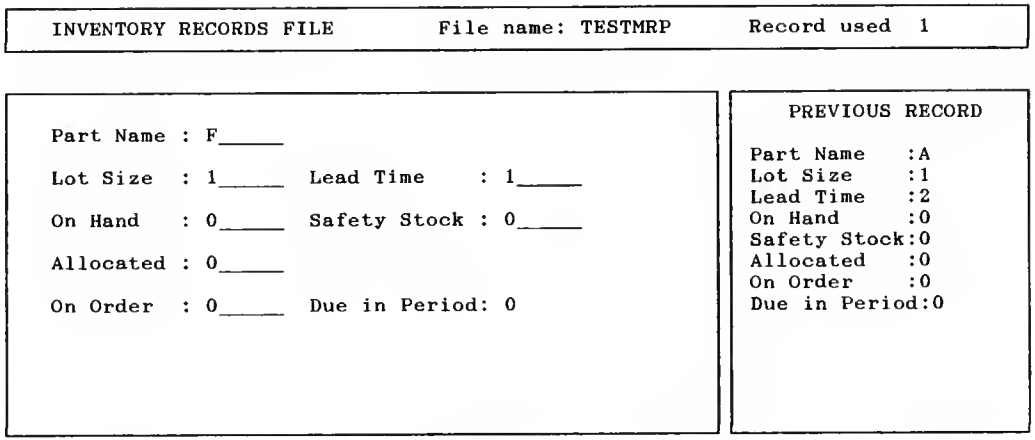
INVENTORY RECORDS FILE File name: TESTMRP Record used 1
Part Name : F_____
Lot Size : 1_____ Lead Time : 1_____
On Hand : 0_____ Safety Stock : 0_____
Allocated : 0_____
On Order : 0_____ Due in Period: 0
PREVIOUS RECORD
Part Name :A
Lot Size :1
Lead Time :2
On Hand :0
Safety Stock:0
Allocated :0
On Order :0
Due in Period:0
Command ? : A)dd D)elete Q)uit:

Figure 10 An example of entering part number F into the Inventory Records File

BILL OF MATERIAL RELATIONSHIPS FILE

The Bill of Material Relationships file is used to keep the record of the relationships between the part made and the sub-parts required to make the parent part, and the quantity of sub-parts per made part. In order to create and organize the Bill of Material Relationships file, the program call "DATA2.BOM" consists of six procedures (procedures Database, Makescreen, Ebom, Bclear, Badd, and Bdelete) need to be used see Figure 11 for Bill of Material Relationships file block diagram. Beginning at procedure Database (see APPENDIX A-10 and B-10), this procedure is used to create an MRP database menu screen option see Figure 5 for MRP database screen. When update BOM Relationships option is selected the procedure called procedure Makescreen, see APPENDIX A-8 and B-8,for making a CRT frame, this is the same frame that we used in the Part List edit screen, and then procedure Ebom, see APPENDIX A-3 and B-3, creates or deletes records containg the required BOM information. See Figure 12 for Bill of Material Relationships file format screen. From here if the add record option has been selected the procedure Badd, see APPENDIX A-8 and B-8, will create a new data record. The procedure Badd creates a new code number by joining Part make number and Part needed number together (by concatenation of Part make number and Part needed number for example Part make number "A" and Part needed number "B" so new code number is "AB".) as a key index when the new

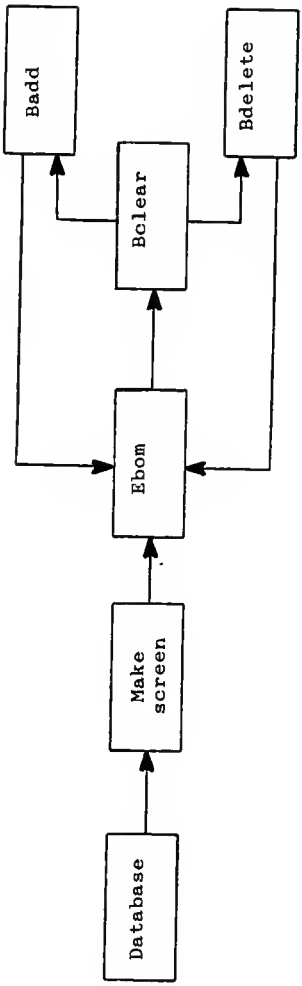

Figure 11 The BOM Relationship procedure block diagram

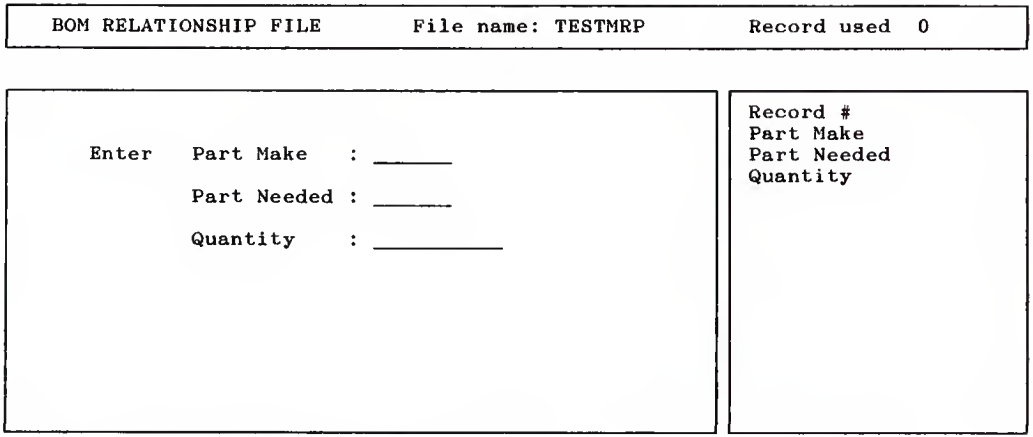

Figure 12 BOM Relationship File edit screen

record has been entered, the procedure Badd needs to check if the new code number is already assigned or not. If the new code number is not assigned, the procedure will add a new data record into BOM Relationships data file (information associated with code number will keep in this file) and BOM Relationships index file (code number as a key index will be kept in this file). If the delete record option has been selected the procedure Bdelete, see APPENDIX A-8 and B-8, will create the proper search code to delete the desired record. After each add or delete has been performed the program will loop back to procedure Ebom, for the next add or delete record or return to the main program. The descriptive pseudo code for the procedure add and delete in Bill of Material Relationships file are given in Figures 13 and 14.

PROCEDURE TO ADD BOM RELATIONSHIP RECORDS

This procedure is used to add the records (part made, part needed and quantity of part needed) into the BOM data file. The procedure also adds three indexes (part made, part needed and code number) into the part made index file, part needed index file and code number index file. Code number is created to show the relationship between part made and part needed.

```
DECLARE  part made, part needed, code number, answer
           AS STRING
         quantity AS INTEGER


BEGIN
  open bom data file
  open part made index file
  open part needed index file
  open code number index file
  REPEAT UNTIL answer = no
    enter part made to be add
    enter part needed associated with part made
    code number = part needed + part made
    set field code number = code number
    IF code number has been found in the code number
       index file
      THEN BEGIN
        enter quantity of part needed
        add data record into the BOM data file
        add code number into the code number index
        file
        add part made into the part made index file

        add part needed into the part needed index
        file
          END
      ELSE
        print already assigned code number
    ENDIF
  ENDREPEAT
  close BOM data file
  close code number index file
  close part made index file
  close part needed index file
END
```

Figure 13 Procedure To Add BOM Relationship Records

PROCEDURE TO DELETE BOM RELATIONSHIP RECORDS

```
This procedure is used to delete BOM records and
BOM index records. The procedure uses part made and
part needed to make code number. This code number is
a key for searching in the code number index file, if
code number is found in the code number index file,
the procedure will delete the BOM records and BOM
index records.

DECLARE  part made, part needed, code number, answer
           AS STRING


BEGIN
  open bom data file
  open part made index file
  open part needed index file
  open code number index file
  REPEAT UNTIL answer = no
    enter part made to be deleted
    enter part needed associated with part made
    code number = part needed + part made
    set field code number = code number
    delete code number in the code number index file


    IF procedure delete has been a success
      THEN BEGIN
         delete part made in the part made index file
         delete part needed in the part needed index
           file
         delete BOM data record in the BOM data file
           END
      ELSE
        print code number to be delete not found
    ENDIF
  ENDREPEAT
  close BOM data file
  close code number index file
  close part made index file
  close part needed index file
END
```

Figure 14 Procedure To Delete BOM Relationship Records

NUMERICAL EXAMPLE FOR BOM RELATIONSHIPS DATABASE FILE

To illustrate how to enter the data into BOM Relationships file. The data from Figure 1 will considered. Part make number, Part Needed number, and Quality of Part needed to make Part make needed to be prepared for data entries. Call the program MRP and follow the menu instruction. Figures 15 and 16 are shown the example for entering Part Make "A" and Part Needed "B", Part Make "A" and Part Needed "C" into the BOM Relationship file.

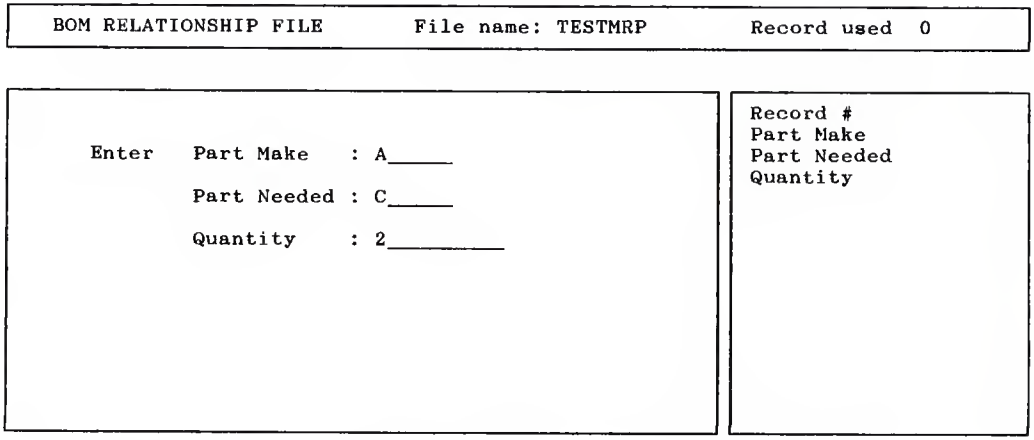

Figure 15 An example for entering Part Make A and Part Needed C into the BOM Relationship File

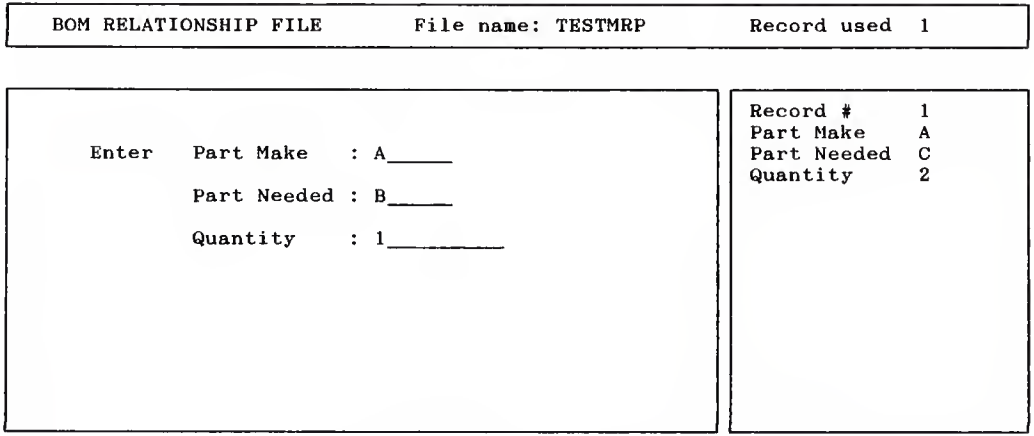
BOM RELATIONSHIP FILE File name: TESTMRP Record used 1
Enter Part Make : A_____
Part Needed : B_____
Quantity : 1_____
Record # 1
Part Make A
Part Needed C
Quantity 2
Command ? : A)dd D)elete Q)uit:

Figure 16 An example for entering Part Make A and Part Needed B into the BOM Relationship File

MASTER PRODUCTION SCHEDULE FILE

The Master Production Schedule file is used to keep the MRP independent demand data (Master Schedule input) for end products or finish items. See Figure 3 for independent demand requirements. In order to create and organize the Master Production Schedule data and index files, the program called "DATA3.MS" consists of six procedures (procedure Database, Makescreen, Master, Mdel, Maddp, and Mdelete) see Figure 17 for Master Production Schedule file block diagram and also see APPENDIX B-5, B-8, B-10 for the pseudocode of each procedure. The Master Production Schedule file uses Part number as the index key to find the record information. The same for Part List file and BOM file. Two main procedures add and delete are used to organize the Master Production Schedule file. The pseudocode add and delete are shown in Figures 18 and 19.

NUMERICAL EXAMPLE FOR MASTER PRODUCTION SCHEDULE FILE

To illustrate how to enter the data into the Master Production Schedule records file. The data from Figure 3 will considered. Call the program MRP and follow the menu instruction. Figures 20, 21, and 22 are shown the example for entering Part "A", "F", and "C" into the Master Production Schedule file.

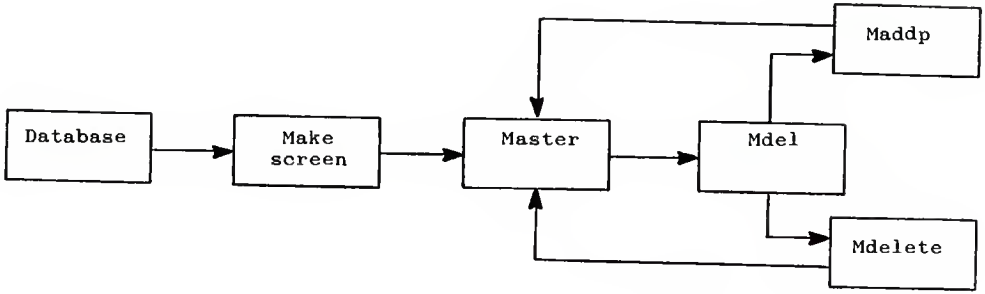

Figure 17 The Master Production Schedule procedure block diagram

```
PROCEDURE TO ADD INFORMATION TO THE MASTER PRODUCTION
        SCHEDULE

     This procedure is used to add informations such as
     part number and independent demands for each
     period into the master production schedule file.
     The procedure uses part number as a key index.


     DECLARE  part number, answer AS STRING
              independent demands for each period  AS
              INTEGER



     BEGIN
       open master production schedule data file
       open master production schedule index file
       REPEAT UNTIL answer = no
         enter part number to be added
         find part number in the master production
           schedule index file
         IF part number has been found in the index
            file
            THEN BEGIN
              enter independent demands for each period
              add data record into the master
                production schedule data file
              add part number into the master
                production schedule index file
                 END
         ELSE
           print already assigned part number
         ENDIF
       ENDREPEAT
       close master production schedule data file
       close master production schedule index file
     END
```

Figure 18 Procedure To Add Information to the Master Production Schedule Records

PROCEDURE TO DELETE MASTER PRODUCTION SCHEDULE RECORDS

This procedure is used to delete the master production schedule records. The procedure uses part number as a key for searching in the master production schedule index file. If part number has been found in the index file then data record associated with part number have been deleted.

```
DECLARE  part number, answer AS STRING

BEGIN

  open master production schedule data file
  open master production schedule index file
  REPEAT UNTIL answer = no
    enter part number to be delete
    delete part number from the master production
      schedule index file

    IF procedure delete has been success
       THEN BEGIN
         delete master production record in the
           master production schedule data file
             END
       ELSE
         print part number to be delete not found
    ENDIF
  ENDREPEAT
  close master production schedule data file
  close master production schedule index file
END
```

Figure 19 Procedure To Delete Master Production Schedule Records

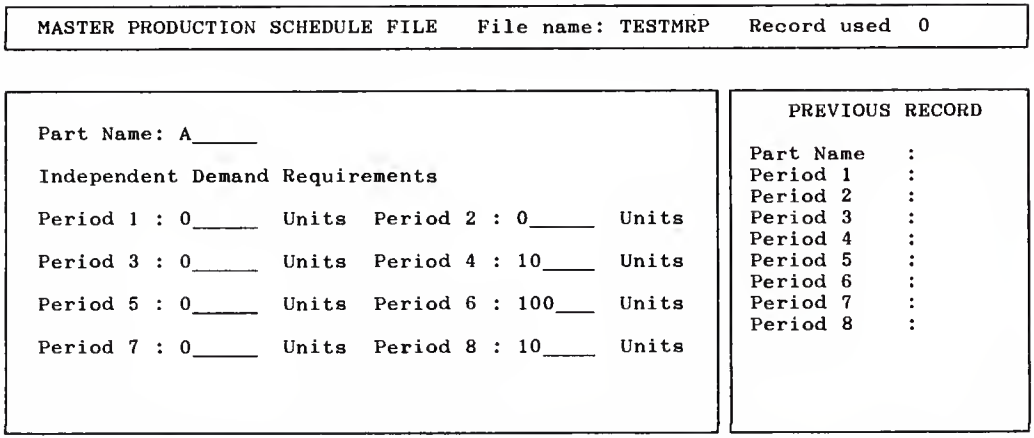
MASTER PRODUCTION SCHEDULE FILE File name: TESTMRP Record used 0
Part Name: A_____
Independent Demand Requirements
Period 1 : 0_____ Units Period 2 : 0_____ Units
Period 3 : 0_____ Units Period 4 : 10____ Units
Period 5 : 0_____ Units Period 6 : 100___ Units
Period 7 : 0_____ Units Period 8 : 10____ Units
PREVIOUS RECORD
Part Name :
Period 1 :
Period 2 :
Period 3 :
Period 4 :
Period 5 :
Period 6 :
Period 7 :
Period 8 :
Command ? : A)dd D)elete Q)uit:

Figure 20 An example for entering Part A into the Master Production Schedule File

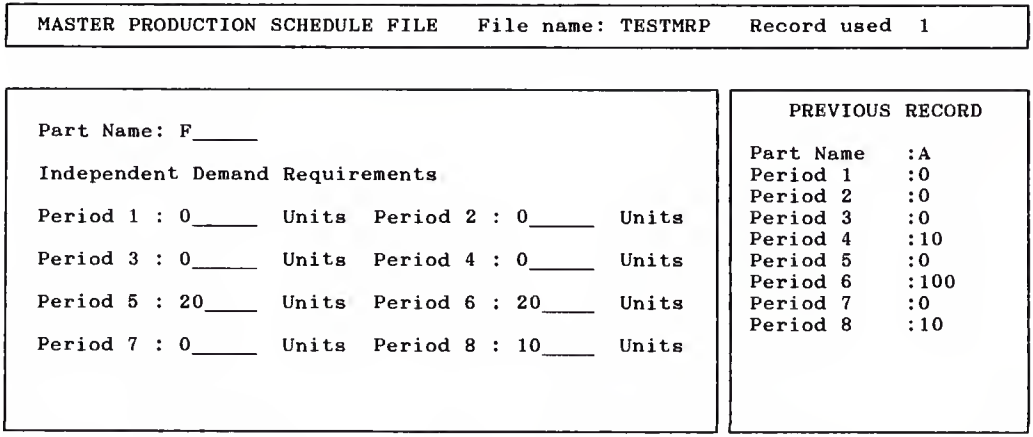
MASTER PRODUCTION SCHEDULE FILE File name: TESTMRP Record used 1
Part Name: F_____
Independent Demand Requirements
Period 1 : 0_____ Units Period 2 : 0_____ Units
Period 3 : 0_____ Units Period 4 : 0_____ Units
Period 5 : 20____ Units Period 6 : 20____ Units
Period 7 : 0_____ Units Period 8 : 10____ Units
PREVIOUS RECORD
Part Name :A
Period 1 :0
Period 2 :0
Period 3 :0
Period 4 :10
Period 5 :0
Period 6 :100
Period 7 :0
Period 8 :10
Command ? : A)dd D)elete Q)uit:

Figure 21 An example for entering Part F into the Master Production Schedule File

LEVEL CODE ASSIGNMENT PROCEDURE

Before MRP explosion processing, each part number is assigned a level number which indicates its dependency relationship with the other part numbers. L. E. Grosh introduced a technique to assign the level code. The procedure requires the use of the parts list and a BOM relationships list sorted on part made major. The steps of the procedure are as follows:

1. Set level code (L) = 1.
2. Identify the parts in the parts list that do not appear in the parts used column of the BOM list. These parts belong to this level.
3. Working with the parts assigned to this level, remove all BOM entires with these part needed numbers.
4. Recopy the reduced list.
5. Proceed to the next level i.e. L = L + 1 (back to step 1) if neither list is empty otherwise stop.

This procedure was applied to the our MRP programs under subprogram call "ASSIGN.LV". The subprogram "ASSIGN.LV" consists of four procedures (procedures Copy, Filecopy, Assignlv, and Codefile) see Figure 23 for assigned level code procedure block diagram and see APPENDIX B-4 for the pseudocode.

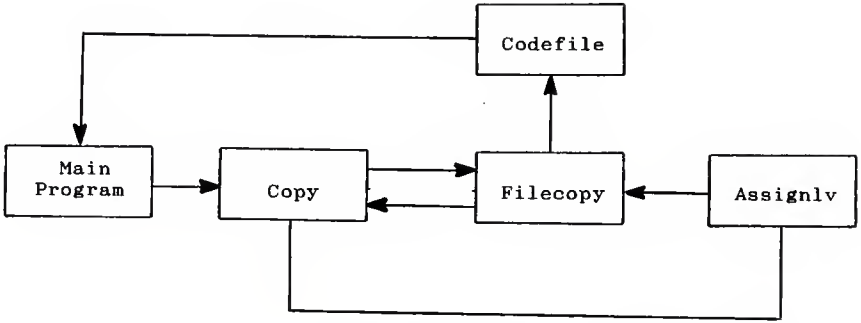

Figure 23 The BOM Assigned Level Code procedure block diagram

NUMERICAL EXAMPLE FOR ASSIGNED LEVEL CODE PROCEDURE

To illustrate assigned level code procedure, the example data from Figures 1, 2, and 3 will considered. Figure 24 is shown the procedure assigned level code, the procedure was performed by hand. Consider the same example data, by using MRP program assume that all of data have been entry into Inventory record file, BOM Relationships file, and Master Production Schedule file. The result from the computer is shown in Figure 25.

Part List	BOM couplete: Part Made	BOM couplete: Part Needed
A *	A	C **
B	A	B **
C	A	D **
D	F	B **
E	F	D **
F *	F	E **
	B	C
	B	E
Level code 1: A, F		

Part List	BOM couplete: Part Made	BOM couplete: Part Needed
B *	B	C **
C	B	E **
D		
E *		
Level code 2: B, E		

Part List	BOM couplete: Part Made	BOM couplete: Part Needed
C		
D		
Level code 3: C, D		

Note : * Part number not in part made number list

** BOM couplete to be eliminated

Figure 24 An example of the assigned level code procedure

```
BOM LEVEL CODES PROCEDURE
-------------------------

LEVEL 1

  a  f

LEVEL 2

  b  d

LEVEL 3

  c  e

Level Code Procedure Completed ... Please hit Enter Key
```

Figure 25 The computer output from procedure BOM assigned level code

MRP EXPLOSION PROCEDURE

The techniques of an MRP explosion that we have used in our MRP programs, is introduced by the APICS training aid.[22] After a Level Code procedure is completed, all information assosiated with Part Number are stored in the temporary files. The temporary files will used to perform the MRP explosion. For each part number the MRP explosion process consists of the following steps:

Net Requirements and Planned orders by level code

Step 1. while i = 1

Get all of independent demand from the Master Production Schedule records file associated with lower level code and then place the quantities of demands into their appropriated time period in gross requirement of each part number.

while i > 1

To develop the gross requirements for level i > 1. Sum the Planned order releases for level j = 1, 2, .., i - 1 multiplied by the BOM quantity per part made.

Step 2.

Get the necessary Inventory Data (lot size, lead time, on hand, safety stock, and allocated) associated with each part number and place into MRP profile.

Step 3.

Calculate the net requirement. "Net requirement are determined by adding allocations to gross requirements and then reducing gross requirements (on a period-by-period basis) by available on-order inventory (scheduled receipts) and projected on-hand inventory."[23]. Calculation performs for each period (period 1 thorough period 8).

Step 4.

Determine Planned order receipts for each period by copying net requirements when a part's requirements are not "covered" by projected on hand inventory.

Step 5.

Determine Planned order releases for each period by looking at Planned order receipts data, but Planned order releases must be offset by the part its lead time. This Planned order releases is a look ahead or anticipated order to cover anticipated further needs.

In the MRP program, the MRP explosion procedure is in the subprogram call "MRP.P". This subprogram consists of four procedures (procedures Computing_MRP, First_level, Net_rq, and Net_lfl) see Figure 26 for the MRP explosion procedure block diagram and also see APPENDIX B-1 for pseudocode. The procedure Computing_MRP is used to prepare

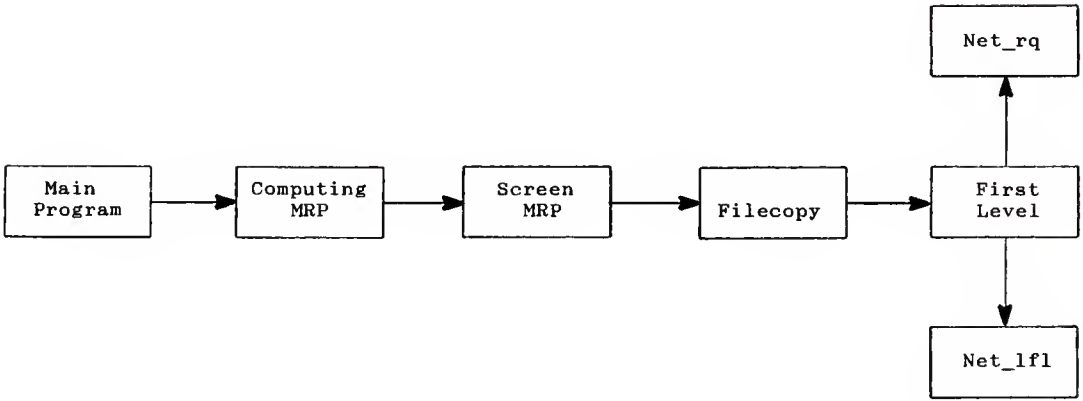

Figure 26 The MRP explosion procedure block diagram

results files for keeping the final MRP results. This procedure loops through all level codes. The procedure First_level is used to perform the MRP explosion calculations and then create MRP results on the CRT, those results are kept in the MRP data file and by using part number as an index key in the MRP results file. The procedures Net_rq and Net_lfl are used to compute net requirements, Projected On Hand, and Planned Order Receipts. The Net_rq procedure will perform while Lot Size policy is not equal to Lot-for-Lot, if Lot Size policy is Lot-for-Lot type then the Net_lfl procedure will be used.

NUMERICAL EXAMPLE FOR MRP EXPLOSION PROCEDURE

To illustrate procedure MRP explosion the example data from Figure 1, 2, and 3 will considered again. Assume that assigned level code procedure for each part numbers must be completed. Call the program MRP and follow the menu instruction. The results for the example is shown in Figures 27, 28, 29, 30, 31, and 32.

STOCK STATUS					MRP PLANNING
Level Code	: 1	Safety Stock	:	0	
Lot Size	: LFL	Allocated	:	0	ITEM NUMBER : A
Lead Time	: 2	On Order	:	0	
On Hand	: 0	Due in Period	:	NONE	

Period	1	2	3	4	5	6	7	8
Gross Requirements				10		100		10
Scheduled Receipts								
Projected On Hand								
Net Requirememts				10		100		10
Planned Order Receipts				10		100		10
Planned Order Releases		10		100		10		

Please hit enter key for result

Figure 27 The MRP result of part number A

STOCK STATUS					MRP PLANNING
Level Code	: 1	Safety Stock	: 0		
Lot Size	: LFL	Allocated	: 0	ITEM NUMBER :	F
Lead Time	: 1	On Order	: 0		
On Hand	: 0	Due in Period	: NONE		

Period	1	2	3	4	5	6	7	8
Gross Requirements					20	20		10
Scheduled Receipts								
Projected On Hand								
Net Requirememts					20	20		10
Planned Order Receipts					20	20		10
Planned Order Releases				20	20		10	

Please hit enter key for result

Figure 28 The MRP result of part number F

STOCK STATUS					MRP PLANNING	
Level Code	: 2	Safety Stock	:	0		
Lot Size	: LFL	Allocated	:	0	ITEM NUMBER :	B
Lead Time	: 1	On Order	:	0		
On Hand	: 100	Due in Period	:	NONE		

Period	1	2	3	4	5	6	7	8
Gross Requirements		10		120	20	10	10	
Scheduled Receipts								
Projected On Hand	100	90	90					
Net Requirememts				30	20	10	10	
Planned Order Receipts				30	20	10	10	
Planned Order Releases			30	20	10	10		

Please hit enter key for result

Figure 29 The MRP result of part number B

STOCK STATUS					MRP PLANNING
Level Code	: 2	Safety Stock	: 0		
Lot Size	: 160	Allocated	: 120	ITEM NUMBER :	D
Lead Time	: 1	On Order	: 0		
On Hand	: 170	Due in Period	: 1		

Period	1	2	3	4	5	6	7	8
Gross Requirements		20		220	20	20	10	
Scheduled Receipts	100							
Projected On Hand	150	130	130	70	50	30	20	20
Net Requirememts				90				
Planned Order Receipts				160				
Planned Order Releases			160					

Please hit enter key for result

Figure 30 The MRP result of part number D

STOCK STATUS					MRP PLANNING
Level Code	: 3	Safety Stock	: 15		
Lot Size	: 150	Allocated	: 0	ITEM NUMBER :	C
Lead Time	: 2	On Order	: 0		
On Hand	: 120	Due in Period	: NONE		

Period	1	2	3	4	5	6	7	8
Gross Requirements	10	30	70	250	30	50	10	10
Scheduled Receipts								
Projected On Hand	110	80	160	60	30	130	120	110
Net Requirememts			5	105		35		
Planned Order Receipts			150	150		150		
Planned Order Releases	150	150		150				

Please hit enter key for result

Figure 31 The MRP result of part number C

STOCK STATUS					MRP PLANNING
Level Code	: 3	Safety Stock	:	0	
Lot Size	: 140	Allocated	:	0	ITEM NUMBER : E
Lead Time	: 1	On Order	:	0	
On Hand	: 120	Due in Period	:	NONE	

Period	1	2	3	4	5	6	7	8
Gross Requirements			90	80	50	30	10	
Scheduled Receipts								
Projected On Hand	120	120	30	90	40	10		
Net Requirememts				50				
Planned Order Receipts				140				
Planned Order Releases			140					

Computing MRP Completed ..hit enter key

Figure 32 The MRP result of part number E

PROGRAM LIMITATIONS

The program will presently handle up to 1000 records (the number of records that can be stored in the Inventory Record file, BOM Relationship file, and Master Production Schedule file). To create a larger number of records, the constant "partregsize" in the MRP.PAS program should be increased. See APPENDIX A-10 for program debugging. The maximum number of records depend upon the capacity of available disk space.

SPEED OF THE MRP PROGRAM

The speed of the MRP program depends upon the types of microprocessor or CPU that are used in the microcomputers, and the types of disk that are used with the microcomputer. Three kinds of disk (floppy disk, hard disk, and RAMdisk) are used for comparison the speed of MRP program. For testing the performance time of the program, two procedures are considered, by testing at procedure BOM assigned level code and by testing at procedure MRP explosion. The following equipments are used in this test:

1. The IBM XT with 8088 CPU at 4.77 MHz. speed.
2. The AT&T 6300 with 8086-2 at 8 MHz. speed.
3. The AT&T 6300 with NEC-V30 CPU at 8 MHz. speed (NEC-V30 is a 8086 compatible microprocessor).
4. The IBM AT with 80286 CPU at 6 MHz. speed.
5. The IBM AT with 80286 CPU at 8 MHz. speed.

6. A 360KB Floppy disk.
7. A 1.2MB Floppy disk.
8. 20MB Hard disk at 65ms access time.
9. 2MB Memory Expansion Board by using 256KB chips at 170ns access time installed for creating Ramdisk.
10. 384KB RAMdisk at 150ns access time in the main memory created by Vdisk program from DOS.

The results of the performance time (using data from the example) at procedure BOM assigned level code is shown in Figure 33. And the results of the performance time at procedure MRP explosion is shown in Figure 34.

Disk Types	CPU type Average Performance Time in sec. 8088 (4.77MHz)	8086-2 (8MHz)	NEC V30 (8MHz)	80286 (6MHz)	80286 (8MHz)
360KB Floppy Disk	64.1	26.1	23.7	54.5	31.6
1.2MB Floppy Disk	-	-	-	32.1	-
20MB Hard Disk (65ms)	56.4	17.5	18.3	9.0	18.4
2MB Expandsion Memory (170ns)	41.4	1.8	1.5	-	-
384KB RAMdisk (150ns)	3.9	2.1	1.7	1.5	1.0

Note : Testing the speed used three different computers.
IBM XT (8088 CPU)
AT&T 6300 (8086-2 and NEC-V30 CPU)
IBM AT (80286 CPU)

Figure 33 Testing the speed of the MRP program at procedure BOM assigned level code (performance time used to finish the procedure).

Disk Types	CPU type Average Performance Time in sec. 8088 (4.77MHz)	8086-2 (8MHz)	NEC-V30 (8MHz)	80286 (6MHz)	80286 (8MHz)
360KB Floppy Disk	18.1	9.8	9.3	20.5	9.7
1.2MB Floppy Disk	-	-	-	12.6	-
20MB Hard Disk (65ms)	14.7	6.7	6.5	3.8	6.6
2MB Expandsion Memory (170ns)	10.6	0.9	0.8	-	-
384KB RAMdisk (150ns)	1.6	0.9	0.8	0.9	0.7

Note : Testing the speed used three different computers.
IBM XT (8088 CPU)
AT&T 6300 (8086-2 and NEC-V30 CPU)
IBM AT (80286 CPU)

Figure 34 Testing the speed of the MRP program at procedure MRP explosion (performance time used to finish the calculation result of one part number).

CONCLUSION

The objectives of this thesis

The following conclusions of this study can be made:

1. This MRP program written in the Turbo Pascal language, has the capability to run only in the microcomputers under PC-DOS or MS-DOS operating system. The program can not be run on the larger computers such as mini or mainframe computers because of the Turbo Pascal language is not the Pascal language, some of the commands have been written only for the Turbo Pascal language.

2. The limit of storing data records is dependent upon the capabilities of disk drives that are used in the microcomputers. This MRP program needs at least double the space used to contain the data records, because the program uses back up data records for calculations only, this is to protect raw data records.

3. The speed of the MRP program is depend upon the type and speed of disk drives that are used to keep data records and the amount of data that is used to calculate. To solve small problems, RAMdisk is recommended because it gives the fastest speed for performing the MRP calculation.

APPENDIX A

PROGRAMS (DEBUGGING)

APPENDIX A-1

PROGRAM (DEBUGGING) CALL "MRP.P"

```
procedure net_lfl(var ng,nal,nsr,noh,nn,nh : integer);
(* This procedure is used to compute net requirement for
   lot-for-lot. *)

begin
 if 0 = noh then
   begin  (* calculate  net requirements and projected on
             hands, when projected on hand equal to 0 *)
     nn := ng + nal - nsr - noh;
     nh := 0;
   end
 else
   begin  (* calculate net requirements and projected on
             hands, when projected on hand not equal to
             0 *)
     if noh > ng then
       begin
         nh := noh - ng;
         nn := 0;
       end
     else
       begin
         nn := ng - noh;
         nh := 0;
       end;
   end;
end;



procedure net_rq(var nls,ng,nal,nsr,noh,nss,d,nn,nh,nor :
                        integer);
(* This procedure is used to compute net requirement. *)
begin
  if (d = 1) then
    begin
      nh := abs((ng + nal) - nsr - noh);
      nn := 0;
      nor := nn;
    end
  else
    begin
      nh := noh - ng;
      if (nss < nh) or (nh = 0) then
        begin
          nh := abs(nh);
          nn := 0;
          nor := nn;
        end
      else
```

```
        begin
          nn := abs(nh - nss);
          nor := nls;
          nh := nor + nh;
        end;
      end;
    end;



procedure first_level(var a : integer);
(* This procedure is used to perfrom the first level of MRP
   calculation. *)

type
  mr = array[1..8] of integer;

var
  g,nm,rp,n,h,orr,ol,nol,srr  : mr;
  b1,c,d,e,lc,al,ss,oh,lt,ls,sr,dp,
  c1,d1,control,q,s1,newoh          : integer;
  an                                : char;
  pn,pneed,pmade                    : string[6];

begin
  clearkey(w2file);
  for b1 := 1 to usedrecs(w2datf) do
  begin
    nextkey(w2file,datref,c);  (* enter assigned level code
                                  number from index file
                                  call "MRP.INX" *)
    if a = c then
    begin   (* for assigned level code = 1 *)
      getrec(w2datf,datref,r1); (* get informations from
                                   data record file call
                                   "MRP.DAT" *)
      pn := r1.partname;
      lc := r1.levelcode;
      al := r1.allocate;
      ss := r1.safety_stock;
      oh := r1.on_hand;
      lt := r1.lead_time;
      ls := r1.lot_size;
      sr := r1.on_order;
      dp := r1.d_period;
      newoh := oh;

      findkey(nmfile,datref,pn);  (* find part number
                                     from index file *)
      if ok then
        begin
          getrec(nmdatf,datref,m);  (* get informations
                                       from data file *)
```

```
g[1] := m.p1;
g[2] := m.p2;
g[3] := m.p3;
g[4] := m.p4;
g[5] := m.p5;
g[6] := m.p6;
g[7] := m.p7;
g[8] := m.p8;
if (1 = ls) then
   begin          (* for lot-for-lot type *)
     for d := 1 to 8 do
     begin (* compute net requirements,
              projected on hand, planned order
              receipts, and scheduled recipts *)

       net_lfl(g[d],al,sr,oh,n[d],h[d]);
       oh := h[d]; orr[d] := n[d];
       srr[d] := sr;
     end;
   end
 else
   begin  (* for not lot-for-lot type *)
     for d := 1 to 8 do
     begin  (* compute net requirements,
               projected on hand, planned order
               receipts, and scheduled recipts *)

     srr[d] := sr;
     net_rq(ls,g[d],al,sr,oh,ss,d,n[d],h[d],
            orr[d]);
     oh := h[d];
     sr := 0;
   end;
 end;
 for d := 1 to 8 do
 begin
   ol[d] := 0; (* set all planned order
                  releases = 0 *)
 end;
 for d := 8 downto 1 do
 begin
   e := d - lt;
   if e >= 1 then
     begin
       ol[e] := orr[d]; (* set planned order
                           releases equal to
                           planned order
                           receipts *)
     end
   else
   end;

 (* write MRP results on screen *)

 textcolor(15);
```

```
gotoxy(1,25);
write('Please hit enter key for result ');
read(an);
gotoxy(17,3); write(lc:6);  (* write assigned
                               level code *)
if ls = 1 then
  begin
    gotoxy(17,4); write('LFL':6); (* write lot
                                     size *)
  end
else
  begin
    gotoxy(17,4); write(ls:6); (* write lot
                                  size *)
  end;
gotoxy(17,5); write(lt:6); (* write lead time *)
gotoxy(17,6); write(newoh:6);(* write on hand *)
gotoxy(42,3); write(ss:6); (* write safety
                              stock *)
gotoxy(42,4); write(al:6); (* write allocated *)
gotoxy(42,5); write(sr:6);  (* write on order *)
if dp = 0 then
  begin
    gotoxy(42,6); write('NONE':6);
  end
else
  begin
    gotoxy(42,6); write(dp:6); (* write time
                                  period *)
  end;
gotoxy(66,4); write(pn:6); (* write part
                              number *)
s1 := 25;
for d := 1 to 8 do
begin
if g[d] <> 0 then
  begin
    gotoxy(s1,12); write(g[d]:6); (* write
                        gross requirements *)
  end
else
  begin
    gotoxy(s1,12); write('      ');
  end;
  s1 := s1 + 7;
end;
s1 := 25;
for d := 1 to 8 do
  begin
    if srr[d] <> 0 then
       begin
         (* write scheduled receipts *)
        gotoxy(s1,14); write(srr[d]:6);
       end
    else
```

```
      begin
        gotoxy(s1,14); write('       ');
      end;
      s1 := s1 + 7;
    end;
s1 := 25;
for d := 1 to 8 do
begin
  if h[d] <> 0 then
    begin
      gotoxy(s1,16); write(h[d]:6); (* write
                                       on hands *)
    end
  else
    begin
      gotoxy(s1,16); write('       ');
    end;
    s1 := s1 + 7;
  end;
  s1 := 25;
  for d := 1 to 8 do
    begin
      if n[d] <> 0 then
        begin
          gotoxy(s1,18); write(n[d]:6); (* write
                                 net requirements *)
        end
      else
        begin
          gotoxy(s1,18); write('       ');
        end;
        s1 := s1 + 7;
    end;
  s1 := 25;
  for d := 1 to 8 do
    begin
      if orr[d] <> 0 then
        begin
          (* write planned order receipts *)
          gotoxy(s1,20); write(orr[d]:6);
        end
      else
        begin
          gotoxy(s1,20); write('       ');
        end;
        s1 := s1 + 7;
    end;
  s1 := 25;
  for d := 1 to 8 do
    begin
      if ol[d] <> 0 then
        begin
          (* write planned order releases *)
          gotoxy(s1,22); write(ol[d]:6);
        end
      else
```

```
          begin
            gotoxy(s1,22); write('       ');
          end;
          s1 := s1 + 7;
      end;
    with mp do
    begin
    (* perpare all MRP results for saving into
       MRP results data/index files *)
    part_name := pn;
    g1 := g[1]; s1 := sr; h1 := h[1];
    n1 := n[1];
    or1 := orr[1]; ol1 := ol[1];
    g2 := g[2]; s2 := sr; h2 := h[2]; n2 := n[2];
    or2 := orr[2]; ol2 := ol[2];
    g3 := g[3]; s3 := sr; h3 := h[3]; n3 := n[3];
    or3 := orr[3]; ol3 := ol[3];
    g4 := g[4]; s4 := sr; h4 := h[4]; n4 := n[4];
    or4 := orr[4]; ol4 := ol[4];
    g5 := g[5]; s5 := sr; h5 := h[5]; n5 := n[5];
    or5 := orr[5]; ol5 := ol[5];
    g6 := g[6]; s6 := sr; h6 := h[6]; n6 := n[6];
    or6 := orr[6]; ol6 := ol[6];
    g7 := g[7]; s7 := sr; h7 := h[7]; n7 := n[7];
    or7 := orr[7]; ol7 := ol[7];
    g8 := g[8]; s8 := sr; h8 := h[8]; n8 := n[8];
    or8 := orr[8]; ol8 := ol[8];
  end;
  findkey(mrpfile,datref,pn); (* find part number
                                 in index file *)
  if ok then
    begin  (* for part number has been assigned *)
      writeln('can not add');
    end
  else
    begin  (* for part number has not been
              assigned *)
      mp.status := 0;
      addrec(mrpdatf,datref,mp); (* add new data
                                    record *)
      (* add new index record *)
      addkey(mrpfile,datref,mp.part_name);
      clearkey(pofile);
      for c1 := 1 to usedrecs(podatf) do
      begin     (* start posting procedure *)
        for d1 := 1 to 8 do
          begin
            nol[d1] := ol[d1];
          end;
        (* enter part made for posting *)
        nextkey(pofile,datref,pmade);
        if pn = pmade then  (* compare part number
                               and part made *)
          begin   (* for part number = part made *)
            getrec(podatf,datref,b);  (* get part
                                         made *)
```

```
pneed := b.pm;
q := b.qty;
findkey(pfile,datref,pneed);
(* find part needed from part list
   index file *)
if ok then
  begin
    (* get informations from part list
       data file *)
    getrec(pdatf,datref,m);

    rp[1] := m.p1; rp[2] := m.p2;
    rp[3] := m.p3; rp[4] := m.p4;
    rp[5] := m.p5; rp[6] := m.p6;
    rp[7] := m.p7; rp[8] := m.p8;

    (* find part needed from MPS index file *)
       findkey(nmfile,datref,pneed);
    if ok then
      begin

        (* get informations *)
        getrec(nmdatf,datref,m);
        nm[1] := m.p1; nm[2] := m.p2;
        nm[3] := m.p3; nm[4] := m.p4;
        nm[5] := m.p5; nm[6] := m.p6;
        nm[7] := m.p7; nm[8] := m.p8;

        for d1 := 1 to 8 do
        begin  (* calculate planned order
                  releases *)
        nol[d1] := (nol[d1] * q) + rp[d1] +
                    nm[d1];
      end;
    end
  else
    begin
      for d1 := 1 to 8 do
        begin  (* calculate planned order
                  releases *)
          nol[d1] := (nol[d1] * q) + rp[d1];
        end;
    end;
  end
else
begin
  findkey(nmfile,datref,pneed); (* find part
                                   needed from
                                   index file *)
  if ok then
  begin
    getrec(nmdatf,datref,m); (* get informations
                                *)
    nm[1] := m.p1; nm[2] := m.p2;
    nm[3] := m.p3; nm[4] := m.p4;
```

```
            nm[5] := m.p5; nm[6] := m.p6;
            nm[7] := m.p7; nm[8] := m.p8;

            for d1 := 1 to 8 do
            begin  (* calculate planned order releases *)
              nol[d1] := (nol[d1] * q) + nm[d1];
            end;
          end
          else
          begin
            for d1 := 1 to 8 do
            begin  (* calculate planned order releases *)
              nol[d1] := nol[d1] * q;
            end;
          end;
        end;
        findkey(pfile,datref,pneed);  (* find part needed
                                         from part list
                                         index file *)
        if ok then
        begin  (* delete part needed data/index record
                  files *)
          deletekey(pfile,datref,pneed);
          deleterec(pdatf,datref);
        end;
        m.status := 0;
        m.partn  := pneed;  (* set part number equal to
                               part needed *)
        m.p1 := nol[1]; m.p2 := nol[2];
        m.p3 := nol[3]; m.p4 := nol[4];
        m.p5 := nol[5]; m.p6 := nol[6];
        m.p7 := nol[7]; m.p8 := nol[8];

        (* add part number into data/index record files *)
        addrec(pdatf,datref,m);
        addkey(pfile,datref,m.partn);
      end;
    end;
  end;
 end
else
end;
end;



procedure computing_mrp;
(* This procedure is used to perfrom MRP calculation. *)

label
  100, 110;
```

```
const
  pdt = '.pdt';
  pnx = '.pnx';
  nmd = '.nmd';
  nmx = '.nmx';

var
  a,c,f1,f2,f3,f4,nc : integer;
  dname              : string[6];
  an : char;

begin
  clrscr;
  screen_mrp;  (* call "SCREEN_MRP" procedure *)

  (* make temporary data/index files for computing MRP *)
  initindex;
  makefile(mrpdatf,'cmrp.dat',partregsize);
  makeindex(mrpfile,'cmrp.inx',6,0);

  (* call back up files procedure *)
  pname := name + pdt;
  pnname := name + pnx;
  nmname := name + nmd;
  nnmname := name +nmx;
  filecopy(bname,pname);
  filecopy(bnname,pnname);
  filecopy(mname,nmname);
  filecopy(msname,nnmname);
  a := 1;

  repeat
   (* open all of data/index files that going to use in MRP
      calculation. *)
    initindex;
    openfile(w2datf,'mrp.dat',partregsize);
    openindex(w2file,'mrp.inx',5,1);
    makefile(pdatf,'pmrp.dat',partregsize);
    makeindex(pfile,'pmrp.inx',6,0);
    openfile(nmdatf,nmname,partregsize);
    openindex(nmfile,nnmname,6,0);
    openfile(podatf,pname,partregsize);
    openindex(pofile,pnname,6,1);

    first_level(a);       (* call "FIRST_LEVEL" procedure *)
    a := a + 1;
    c := a;
    clearkey(w2file);     (* go to the top of records *)
    for f1 := 1 to usedrecs(w2datf) do
      begin
        f3 := usedrecs(w2datf);
        nextkey(w2file,datref,nc); (* enter data record *)
        if c = nc then
```

```
        begin    (* close all data/index files *)
          closefile(podatf);
          closeindex(pofile);
          closefile(nmdatf);
          closeindex(nmfile);
          closefile(pdatf);
          closeidex(pfile);
          closefile(w2datf);
          closeindex(w2file);

          (* make back up copies data/index files *)
          filecopy('pmrp.dat',nmname);
          filecopy('pmrp.inx',nnmname);
          goto 100;
        end
      else
        begin
          if f1 = f3 then
            begin  (* close all data/index files *)
              closefile(w2datf);
              closeindex(w2file);
              closefile(mrpdatf);
              closeindex(mrpfile);
              closefile(pdatf);
              closeindex(pfile);
              closefile(podatf);
              closeindex(pofile);
              c := 0;
              goto 100;
            end
          else
            begin
              goto 110;
            end;
          end;
   110: end;
   100: until c = 0;
        gotoxy(1,25);
        write('Computing MRP Completed ..hit enter key ');
        read(an);
end;
```

APPENDIX A-2

PROGRAM (DEBUGGING) CALL "DATA1.PL"

```
procedure addp;
(* This procedure is used to add part list data/index
   files *)
var
  an : char;
   a : integer;

begin
  textcolor(15);
  gotoxy(17,7); read(pn);  (* enter part number for key
                              broad *)
  p.partl := pn;
  findkey(rfile,datref,pn); (* find part number in part
                               list index file *)
  if ok then
    begin  (* for part number has been assigned *)
      sound(440);
      delay(500);
      nosound;
      for a := 1 to 3 do
        begin
          delay(1500);
          textbackground(12);
          textcolor(15);
          gotoxy(5,19);
          write(' Already assigned part name ');
          delay(1500);
          textbackground(0);
          gotoxy(5,19);
          write('                              ');
        end;
      textcolor(15);
    end
 else
    begin  (* for part number has not been assigned *)
      p.status := 0;
      (* enter all informations associated with part
         number *)
      gotoxy(17,9); read(p.lotsize);
      gotoxy(40,9); read(p.leadtime);
      gotoxy(17,11); read(p.onhand);
      gotoxy(40,11); read(p.safetystock);
      gotoxy(17,13); read(p.allocated);
      gotoxy(17,15); read(p.onorder);
      gotoxy(40,15); read(p.period);

      (* add new data/index record *)
      addrec(datf,datref,p); addkey(rfile,datref,p.partl);

      (* write all informations on screen *)
      gotoxy(72,2); write(usedrecs(datf));
      gotoxy(72,8);  write('      ');
      gotoxy(72,8);  write(pn);
      gotoxy(72,9);  write('      ');
      gotoxy(72,9);  write(p.lotsize);
```

```
      gotoxy(72,10);  write('      ');
      gotoxy(72,10); write(p.leadtime);
      gotoxy(72,11); write('      ');
      gotoxy(72,11); write(p.onhand);
      gotoxy(72,12); write('      ');
      gotoxy(72,12); write(p.safetystock);
      gotoxy(72,13); write('      ');
      gotoxy(72,13); write(p.allocated);
      gotoxy(72,14); write('      ');
      gotoxy(72,14); write(p.onorder);
      gotoxy(73,15); write(' ');
      gotoxy(73,15); write(p.period);
    end;
end;



procedure delete;

(* This procedure is used to delete part list data/index
   records *)
var
  a : integer;
begin
  buflen := 15;
  textcolor(15);
  gotoxy(17,7); read(pn);      (* enter part number to be
                                  delete *)
  deletekey(rfile,datref,pn); (* delete part number index
                                  record *)
  if ok then
    begin
      deleterec(datf,datref);    (* delete part number data
                                    record *)
      gotoxy(72,2); write('        ');
      gotoxy(72,2); write(usedrecs(datf));
    end
  else
    begin  (* for part number not found *)
      textcolor(12);
      sound(440);
      delay(500);
      nosound;
      for a := 1 to 3 do
        begin
          delay(1500);
          textbackground(12); textcolor(15);
          gotoxy(5,19);
          write(' Part name to delete not found ');
          delay(1500);
          textbackground(0);
          gotoxy(5,19);
```

```
          write('                                    ');
        end;
      textcolor(15);
    end;
end;



procedure del;
(* This procedure is used create screen delete option *)
begin
  textcolor(10);
  gotoxy(17,7);  write('_______');
  gotoxy(17,9);  write('_______');
  gotoxy(40,9);  write('_______');
  gotoxy(17,11); write('_______');
  gotoxy(40,11); write('_______');
  gotoxy(17,13); write('_______');
  gotoxy(17,15); write('_______');
  gotoxy(40,15); write('_') ;
end;



procedure update;
(* This procedure is used create screen format for
   inventory records file and create menu for add,
   delete, list, find *)
const
  anw = 'q';
var
  a, an, d, q  : char;
  a1 : integer;
  data : string[6];

begin
  initindex;
  openfile(datf,newname,partregsize); (* open part
                                         list data file *)
  openindex(rfile,rname,6,0); (* open part list index
                                 file *)
  textcolor(10);
  gotoxy(5,2);

  (* create screen format for inventory records file *)

  write('INVENTORY RECORDS FILE        File name: ');
  textcolor(15); write(name);
  textcolor(10);
  gotoxy(59,2); write('Record used ');
  textcolor(15);
  gotoxy(72,2); write(usedrecs(datf));
  textcolor(10);
```

```
        gotoxy(5,7);  write('Part Name : _______');
        gotoxy(5,9);  write('Lot Size  : _______');
        gotoxy(25,9);  write('Lead Time    : _______');
        gotoxy(5,11); write('On Hand   : _______');
        gotoxy(25,11); write('Safety Stock : _______');
        gotoxy(5,13); write('Allocated : _______');
        gotoxy(5,15); write('On Order  : _______');
        gotoxy(25,15); write('Due in Period: _');
        gotoxy(59,6);  write('   PREVIOUS RECORD');
        gotoxy(59,8);  write('Part Name    :');
        gotoxy(59,9);  write('Lot Size     :');
        gotoxy(59,10); write('Lead Time    :');
        gotoxy(59,11); write('On Hand      :');
        gotoxy(59,12); write('Safety Stock:');
        gotoxy(59,13); write('Allocated    :');
        gotoxy(59,14); write('On Order     :');
        gotoxy(59,15); write('Due in Period:');
        gotoxy(5,24);
        write('Command ? : A)dd  D)elete Q)uit: ');
        repeat;
          gotoxy(38,24); read(an);
          gotoxy(38,24); write(' ');
          case an of
            'a' : begin    (* for add part list option *)
                    del;  (* call "DEL" procedure *)
                    addp; (* call "ADDP" procedure *)
                  end;
            'd' : begin       (* for  delete part list option *)
                    del;      (* call "DEL" procedure *)
                    delete;   (* call "DELETE" procedure *)
                  end;
          end;
        until anw = an;
        closefile(datf);     (* close inventory data records file *)
        closeindex(rfile);   (* close inventory index records file *)
end;
```

APPENDIX A-3

PROGRAM (DEBUGGING) CALL "DATA2.BOM"

```
procedure bclear;
(* This procedure is used to erase some parts of BOM
   relationship file screen format *)
begin
  gotoxy(28,8);  write('______');
  gotoxy(28,10); write('______');
  gotoxy(28,12); write('__________');
end;



procedure badd;
(* This procedure is used to add BOM relationship
   data/index files *)
var
  bn,bnn,
  bbn     : string[6];
  q,a1    : integer;
  co      : string[12];

begin
  textcolor(15);
  gotoxy(28,8); read(bn);     (* read part made number for
                                 keybroad *)
  gotoxy(28,10); read(bbn);   (* read part needed number for
                                 keybroad *)
  b.pm    := bn;
  b.pn    := bbn;
  co := bbn + bn;             (* create new code number *)
  b.code := co;
  findkey(bcfile,datref,co); (* find code number in index
                                file *)
  if ok then
    begin          (* for code number has been assigned *)
      sound(440);
      delay(500);
      nosound;
      for a1 := 1 to 3 do
      begin
        delay(1500);
        textbackground(12);
        textcolor(15);
        gotoxy(8,16);
        write(' Already assigned code number ');
        delay(1500);
        textbackground(0);
        gotoxy(8,16);
        write('                              ');
      end;
      textcolor(15);
    end
  else
```

```
    begin    (* for code number has not been assigned *)
    gotoxy(28,12);
    read(q);  (* read quality form keybroad *)
    b.qty := q;

    (* add BOM relationship data/index records *)
    addrec(bdatf,datref,b);
    addkey(bcfile,datref,b.code);
    addkey(bmfile,datref,b.pm);
    addkey(bnfile,datref,b.pn);

    (* write informations on screen *)
    gotoxy(72,2); write('        ');
    gotoxy(72,2); write(usedrecs(bdatf));
    gotoxy(70,6); write('        ');
    gotoxy(70,6); write(usedrecs(bdatf));
    gotoxy(70,7); write('        ');
    gotoxy(70,7); write(bn);
    gotoxy(70,8); write('        ');
    gotoxy(70,8); write(bbn);
    gotoxy(70,9); write('        ');
    gotoxy(70,9); write(q);
  end;
end;



procedure bdelete;
(* This procedure is used to delete BOM relationship
   data/index records *)
var
  bn,bm : string[6];
  co    : string[12];
  a1    : integer;

begin
  buflen := 15;
  textcolor(15);
  gotoxy(28,8); read(bm);      (* read part made for
                                  keybroad *)
  gotoxy(28,10); read(bn);     (* read part needed for
                                  keybroad *)
  co := bn + bm;               (* create code number *)
  deletekey(bcfile,datref,co); (* delete code number from
                                  index file *)
  if ok then
  begin     (* for code number has been found *)
    deletekey(bmfile,datref,bm); (* delete part made index
                                    record *)
    deletekey(bnfile,datref,bn); (* delete part needed
                                    index record *)
    deleterec(bdatf,datref);     (* delete data record *)
```

```
    gotoxy(72,2); write('        ');
    gotoxy(72,2); write(usedrecs(bdatf)); (* write record
                                            number after
                                            deleted *)
  end
else
  begin       (* for code number has not been found *)
    sound(440);
    delay(500);
    nosound;
    for a1 := 1 to 3 do
      begin
        delay(1500);
        textbackground(12);
        textcolor(15);
        gotoxy(8,16);
        write(' Part name to delete not found ');
        delay(1500);
        textbackground(0);
        gotoxy(8,16);
        write('                                ');
      end;
    textcolor(15);
  end;
end;



procedure ebom;
(* This procedure is used to create BOM relationship file
   screen format and menu for add, delete, find, list. *)

const
  anw = 'q';
var
  a, an, d, q  : char;

begin
  initindex;
  (* open BOM relationship data/index files *)
  openfile(bdatf,bname,partregsize);
  openindex(bcfile,bcname,12,0);
  openindex(bmfile,bmname,6,1);
  openindex(bnfile,bnname,6,1);

  (* write BOM relationship screen format *)
  textcolor(10);
  gotoxy(5,2);
  write('BOM RELATIONSHIP FILE       File name: ');
  textcolor(15); write(Name);
  textcolor(10);
  gotoxy(59,2); write('Record used ');
  textcolor(15);
```

```
  gotoxy(72,2); write(usedrecs(bdatf));
  textcolor(10);
  gotoxy(8,8);  write('Enter   Part Make : ______');
  gotoxy(8,10); write('      Part Needed : ______');
  gotoxy(8,12); write('        Quantity  : _________');
  gotoxy(59,6); write('Record # ');
  gotoxy(59,7); write('Part Make ');
  gotoxy(59,8); write('Part Needed ');
  gotoxy(59,9); write('Quantity ');
  textcolor(15);
  repeat;
    textcolor(10);
    gotoxy(5,24);
    write('Command ? : A)dd  D)elete  Q)uit: ');

    read(an);  (* read selected option *)
    gotoxy(38,24); write(' ');
    case an of
      'a' :  begin    (* for add BOM relationship
                         file option *)
               bclear;  (* call "BCLEAR" procedure *)
               badd;    (* call "BADD" procedure *)
             end;
      'd' :  begin    (* for delete BOM relationship file
                         option *)
               bclear;  (* call "BCLEAR" procedure *)
               bdelete; (* call "BDELETE" procedure *)
             end;
    end;
 until anw = an;

 (* close all BOM  relationship data/index files *)
 closefile(bdatf);
 closeindex(bcfile);
 closeindex(bmfile);
 closeindex(bnfile);
end;
```

APPENDIX A-4

PROGRAM (DEBUGGING) CALL "ASSIGN.LV"

```
procedure filecopy(nname,nwname: q);
(*  This procedure is used to back up database files. *)

const
  bufsize = 200;           (* set up buffer size *)
  bufbytesize = 15600;     (* set up buffer byte size *)

var
  source,
  dt     : file;
  buffer : array[1..bufbytesize] of byte;
  norec,
  remaining : integer;

begin
  assign(source, nname);  (* assigned file name to
                             be copy *)
  reset(source);
  assign(dt,nwname);       (* assigned new copy file name *)
  rewrite(dt);
  remaining := filesize(source);
  while remaining > 0 do
  begin
    if bufsize <= remaining then
       norec := bufsize
    else
       norec := remaining;
       blockread(source,buffer,norec); (* read data from
                                          database file *)
       blockwrite(dt,buffer,norec); (* make back up
                                       database file *)
       remaining := remaining - norec;
    end;
    close(dt); (* close back up database file *)
  end;



procedure assignlv;
(*  This procedure is used to do assign level code for each
    part number. *)

label 10,20,30;
const
  use4 = 0;

var
  use1,use2,use3,u1,u2 : integer;
  a,d,c,e,count,count1,count2 : integer;
  data1,data2,data3,data4,data5,data6,data7,data9,
  data10 : string[6];
   data8 : string[12];
      an : char;
       x : integer;
```

```
begin
  clrscr;
  textcolor(14);
  writeln('BOM  LEVEL CODES PROCEDURE');
  writeln('--------------------------');

  (* Create temporary assigned level code data/index
     files call "work1.dat" and "work1.inx" *)
  initindex;
  makefile(w1datf,'work1.dat',partregsize);
  makeindex(w1file,'work1.inx',6,0);
  closefile(w1datf);
  closeindex(w1file);

  count := 0; (* set count assigned level code
                 equal to 0 *)
  x := 0;

  (* get data from part list file and BOM
     relationship file *)
20:  initindex;
  openfile(adatf,aname,partregsize); (* open part list
                                        data file *)
  openindex(arfile,arname,6,0);      (* open part list
                                        index file *)
  makefile(wdatf,'work.dat',partregsize);  (* create
                                              temporary
                                              data file *)
  makeindex(wfile,'work.inx',6,0);   (* create temporary
                                        index file *)
  openfile(w1datf,'work1.dat',partregsize); (* open
                                               temporary
                                               data file *)
  openindex(w1file,'work1.inx',6,0); (* open temporary
                                        index file *)
  use1 := usedrecs(adatf);  (* get number of records
                               for part list *)
  clearkey(arfile);         (* go to top of records *)
  openfile(abdatf,abname,partregsize); (* open
                                          BOM relationship
                                          data file *)
  openindex(abmfile,abmname,6,1); (* open BOM relationship
                                     index file for part
                                     made *)
  openindex(abnfile,abnname,6,1); (* open BOM relationship
                                     index file for part
                                     needed *)
  use2 := usedrecs(abdatf);  (* get number of records for
                                BOM relationship *)
  clearkey(abmfile);         (* go to top of records *)

  if use4 = use2 then
  begin
    textcolor(10);
```

```
  writeln;
  writeln('LEVEL ', count +1);
  count1 := count + 1;
  for e := 1 to use1 do
  begin
    nextkey(arfile,datref,data9);  (* get next data
                                      record *)
    textcolor(15);
    data10 := data9;
    count2 := count1;
    findkey(w1file,datref,data10);
    if ok then
      begin
        writeln('can not add');
      end
    else
      begin
        as.status := 0;
        as.partn := data10;
        as.code  := count2;
        addrec(w1datf,datref,as);
        addkey(w1file,datref,as.partn);
        write('  ',as.partn);
      end;
    end;
    closefile(adatf);
    closeindex(arfile);
    closefile(abdatf);
    closeindex(abnfile);
    closeindex(abmfile);
    closefile(w1datf);
    closeindex(w1file);
    goto 30;
  end;

(* assigned level code procedure start at level 1 *)
textcolor(10);
count := count + 1;       (* set level code *)
writeln;
writeln('LEVEL ',count); (* write level code *)
clearkey(arfile);
for a := 1 to use1 do     (* loop until end of file *)
begin
  clearkey(abmfile);      (* go to top of records *)
  nextkey(arfile,datref,data1);  (* get a part list data
                                    record *)
  data3 := data1;
  findkey(abmfile,datref,data3); (* find part list in
                                    BOM relationship
                                    file *)
  if not ok then
    begin     (* assigned a part list to level code *)
      textcolor(15);
      data10 := data1;
```

```
count2 := count;
p.part1 := data1;

(* keep part list that has been assigned
   level code into temporary files *)
findkey(wfile,datref,data1);
if not ok then
  begin
    p.status := 0;
    addrec(wdatf,datref,p);
    addkey(wfile,datref,p.part1);
    findkey(w1file,datref,data10);
    if ok then
      begin
        writeln('can not add');
      end
    else
      begin
        as.status := 0;
        as.partn := data10;
        as.code  := count2;
        addrec(w1datf,datref,as);
        addkey(w1file,datref,
        as.partn);
        write('  ',as.partn); (* write part
                                 list associated
                                 with the level
                                 code on screen *)
      end;
    end;
  end;
end;

(* delecting part list records have been assigned
   level code in part list data/index files and
   in BOM relationship data/index files. *)
use3 := usedrecs(wdatf);
clearkey(wfile);
for c := 1 to use3 do
  begin
    nextkey(wfile,datref,data4);
    buflen := 15;
    deletekey(arfile,datref,data4);
    deleterec(adatf,datref);
  end;
closefile(adatf);
closeindex(arfile);
clearkey(wfile);
for c := 1 to use3 do
  begin
    nextkey(wfile,datref,data4);
    data5 := data4;
10: findkey(abnfile,datref,data5);
    if ok then
```

```
                begin
                  getrec(abdatf,datref,b);
                  data6 := b.pn;
                  data7 := b.pm;
                  data8 := b.code;
                  deletekey(abnfile,datref,data6);
                  deleterec(abdatf,datref);
                  deletekey(abmfile,datref,data7);
                  goto 10;
                end;
              end;
              closefile(abdatf);
              closeindex(abnfile);
              closeindex(abmfile);
              closefile(w1datf);
              closeindex(w1file);
              writeln;
              goto 20;
          30: textcolor(14);
              writeln;
              writeln;
              write('Level Code Procedure Completed ...
                     Please hit Enter key ');
              readln(an);
end;



procedure codefile;
(* This prodecure is used to prepare informations
   needed for MRP before doing MRP calculation. *)
var
  a1,a2,a3,z4   : integer;
        z1,z2   : string[6];
           an   : char;

begin
  initindex;
  (* open data file call "WORK1.DAT" *)
  openfile(w1datf,'work1.dat',partregsize);

  openindex(w1file,'work1.inx',6,0); (* open index file
                                        call "WORK1.INX" *)
  openfile(adatf,aname,partregsize); (* open part list
                                        data file *)
  openindex(arfile,arname,6,0);      (* open part list
                                        index file *)

  (* create temporary data file *)
  makefile(w2datf,'mrp.dat',partregsize);
```

```
(* create temporary index file *)
makeindex(w2file,'mrp.inx',5,1);
a1 := usedrecs(w1datf);  (* get records number for file
                            "WORK1.DAT" *)
clearkey(w1file);
for a2 := 1 to a1 do
  begin
    nextkey(w1file,datref,z1); (* enter data *)
    if ok then
      begin
        getrec(w1datf,datref,as); (* get informations
                                     associated with data
                                     record for file
                                     "WORK1.DAT" *)
        z2 := as.partn;
        a3 := as.code;
        r1.partname := z2;
        r1.levelcode := a3;

        findkey(arfile,datref,z2);(* find part number
                                     in part list file *)
        if ok then
          begin
            getrec(adatf,datref,p);(* get informations *)
            r1.partname := p.part1;
            r1.levelcode := a3;
            r1.allocate := p.allocated;
            r1.safety_stock := p.safetystock;
            r1.on_hand := p.onhand;
            r1.lead_time := p.leadtime;
            r1.lot_size := p.lotsize;
            r1.on_order := p.onorder;
            r1.d_period := p.period;
            findkey(w2file,datref,z2);(* find part
                                         number in
                                       temporary file *)
            if not ok then
              begin
                r1.status := 1;
                addrec(w2datf,datref,r1);(* add new data
                                            record *)

                (* add new index record *)
                addkey(w2file,datref,r1.levelcode);
              end
            else
              begin
                writeln('can not add data');
              end;
          end
        else
          begin
            writeln('can not find data');
          end;
        end
```

```
        else
          begin
            writeln(' can not get record');
          end;
        end;
        closefile(w1datf);(* close data file "WORK1.DAT" *)
        closeindex(w1file); (* close index file
                               "WORK1.INX" *)

        closefile(w2datf);  (* close temporary data file *)
        closeindex(w2file);(* close temporary index file *)
end;



procedure copy;

(* This procedure is used to create back up files *)
const
  dat = '.dat';
  ixd = '.ixd';
  bom = '.bom';
  bmx = '.bmx';
  bnx = '.bnx';
  adt = '.adt';
  aix = '.aix';
  aom = '.aom';
  abm = '.abm';
  abn = '.abn';

var
  an,y : integer;

begin
  clrscr;
  (* created file name and file type such as data files or
     index files *)
  newname  := name + dat; (* for part list data files *)
  rname    := name + ixd; (* for part list index files *)
  bname    := name + bom; (* for BOM relationship data
                             files *)
  bmname   := name + bmx; (* for BOM relationship index
                             files *)
  bnname   := name + bnx; (* for BOM relationship index
                             files *)

  aname   := name + adt; (* for back up part list
                            data files *)
  arname  := name + aix; (* for back up part list
                            index files *)
  abname  := name + aom; (* for back up BOM relationship
                            data files *)
  abmname := name + abm; (* for back up BOM relationship index
  abnname := name + abn;    files *)
```

```
        (* call file copy procedure *)
        filecopy(newname,aname);  (* back up part list
                                     data file *)
        filecopy(rname,arname);   (* back up part list
                                     index file *)
        filecopy(bname,abname);   (* back up BOM relationship
                                     data file *)
        filecopy(bmname,abmname); (* back up BOM relationship
        filecopy(bnname,abnname);    index files *)


        assignlv;                 (* call assigned level code
                                     procedure *)
        filecopy(newname,aname);  (* back up part list data
                                     file *)
        filecopy(rname,arname);   (* back up part list
                                     index file *)
        codefile;                 (* call codefile procedure *)
end;
```

APPENDIX A-5

PROGRAM (DEBUGGING) CALL "DATA3.MS

```
procedure mdel;
(* This procedure is used to delete some parts of MPS
   screen format file *)
begin
  textcolor(10);
  gotoxy(15,7);   write('______');
  gotoxy(15,11);  write('______');
  gotoxy(41,11);  write('______');
  gotoxy(15,13);  write('______');
  gotoxy(41,13);  write('______');
  gotoxy(15,15);  write('______');
  gotoxy(41,15);  write('______');
  gotoxy(15,17);  write('______');
  gotoxy(41,17);  write('______');
end;


procedure maddp;
(* This procedure is used to add MPS data/index files *)
var
  an : char;
   a : integer;

begin
  textcolor(15);
  gotoxy(15,7); read(pn); (* enter part number from
                             keybroad *)
  m.partn := pn;
  findkey(mfile,datref,pn); (* find part number in
                               MPS index file *)
  if ok then
    begin       (* for part number has been found *)
      sound(440);
      delay(500);
      nosound;
      for a := 1 to 3 do
        begin
          delay(1500);
          textbackground(12);
          textcolor(15);
          gotoxy(10,19);
          write(' Already assigned part name ');
          delay(1500);
          textbackground(0);
          gotoxy(10,19);
          write('                            ');
        end;
      textcolor(15);
    end
  else
    begin     (* for part number has not been found *)
      p.status := 0;
```

```
      (* read all informations associated with part
         number *)
      gotoxy(15,11); read(m.p1);
      gotoxy(41,11); read(m.p2);
      gotoxy(15,13); read(m.p3);
      gotoxy(41,13); read(m.p4);
      gotoxy(15,15); read(m.p5);
      gotoxy(41,15); read(m.p6);
      gotoxy(15,17); read(m.p7);
      gotoxy(41,17); read(m.p8);

      (* add MPS data/index records *)
      addrec(mdatf,datref,m); addkey(mfile,datref,m.partn);

      (* write all informations on screen *)
      gotoxy(72,2);  write(usedrecs(mdatf));
      gotoxy(72,8);  write('      ');
      gotoxy(72,8);  write(pn);
      gotoxy(72,9);  write('      ');
      gotoxy(72,9);  write(m.p1);
      gotoxy(72,10); write('      ');
      gotoxy(72,10); write(m.p2);
      gotoxy(72,11); write('      ');
      gotoxy(72,11); write(m.p3);
      gotoxy(72,12); write('      ');
      gotoxy(72,12); write(m.p4);
      gotoxy(72,13); write('      ');
      gotoxy(72,13); write(m.p5);
      gotoxy(72,14); write('      ');
      gotoxy(72,14); write(m.p6);
      gotoxy(72,15); write('      ');
      gotoxy(72,15); write(m.p7);
      gotoxy(72,16); write('      ');
      gotoxy(72,16); write(m.p8);
    end;
end;



procedure mdelete;
(* This procedure is used to delete MPS data/index records
   file *)
var
  a : integer;
begin
  buflen := 15;
  textcolor(15);
  gotoxy(15,7); read(pn);      (* read part number from
                                  keybroad *)
  deletekey(mfile,datref,pn);(* find and delete part
                                number index record file *)
  if ok then
```

```
    begin       (* for part number has been found *)
      deleterec(mdatf,datref);  (* delete data record *)
      gotoxy(72,2); write('        ');
      gotoxy(72,2);
      write(usedrecs(mdatf)); (* write record number after
                                 deleted *)
    end
  else
    begin       (* for part number has not been found *)
      textcolor(12);
      sound(440);
      delay(500);
      nosound;
      for a := 1 to 3 do
        begin
          delay(1500);
          textbackground(12); textcolor(15);
          gotoxy(10,19);
          write(' Part name to delete not found ');
          delay(1500);
          textbackground(0);
          gotoxy(10,19);
          write('                                ');
        end;
      textcolor(15);
    end;
end;



procedure master;
(* This procedure is used to create master production
   schedule file screen format and menu for add, delete,
   find, list *)
const
  anw = 'q';
var
  an : char;

begin
  initindex;
  openfile(mdatf,mname,partregsize);  (* open MPS data
                                         file *)
  openindex(mfile,msname,6,0);        (* open MPS index
                                         file *)
  textcolor(10);

  (* write MPS screen format *)
  gotoxy(4,2);
  write('MASTER PRODUCTION SCHEDULE FILE   File name: ');
  textcolor(15); write(name);
  textcolor(10);
  gotoxy(59,2); write('Record used ');
```

```
        textcolor(15);
        gotoxy(72,2); write(usedrecs(mdatf));
        textcolor(10);
        gotoxy(4,7);   write('Part Name: ______');
        gotoxy(4,9);   write('Independent Demand Requirements');
        gotoxy(4,11);  write('Period 1 : ______ Units');
        gotoxy(30,11); write('Period 2 : ______ Units');
        gotoxy(4,13);  write('Period 3 : ______ Units');
        gotoxy(30,13); write('Period 4 : ______ Units');
        gotoxy(4,15);  write('Period 5 : ______ Units');
        gotoxy(30,15); write('Period 6 : ______ Units');
        gotoxy(4,17);  write('Period 7 : ______ Units');
        gotoxy(30,17); write('Period 8 : ______ Units');
        gotoxy(59,6);  write('   PREVIOUS RECORD');
        gotoxy(59,8);  write('Part Name   :');
        gotoxy(59,9);  write('Period 1    :');
        gotoxy(59,10); write('Period 2    :');
        gotoxy(59,11); write('Period 3    :');
        gotoxy(59,12); write('Period 4    :');
        gotoxy(59,13); write('Period 5    :');
        gotoxy(59,14); write('Period 6    :');
        gotoxy(59,15); write('Period 7    :');
        gotoxy(59,16); write('Period 8    :');
        gotoxy(5,24);
        write('Command ? : A)dd  D)elete  Q)uit: ');
        repeat;
          gotoxy(38,24); read(an); (* read selected option *)
          gotoxy(38,24); write(' ');
          case an of
            'a' : begin       (* for add MPS option *)
                    mdel;     (* call "MDEL" procedure *)
                    maddp;    (* call "MADDP" procedure *)
                  end;
            'd' : begin       (* for delete MPS option *)
                    mdel;     (* call "MDEL" procedure *)
                    mdelete; (* call "MDELETE" procedure *)
                  end;
          end;
        until anw = an;

        (* close MPS data/index files *)
        closefile(mdatf);
        closeindex(mfile);
end;
```

APPENDIX A-6

PROGRAM (DEBUGGING) CALL "DIRLIST.U"

```
procedure checkfile;
(* This procedure is used to check error data/index
   files and also for creating new data/index files *)
const
  dat = '.dat';
  ixd = '.ixd';

  bom = '.bom';
  bcx = '.bcx';
  bmx = '.bmx';
  bnx = '.bnx';

  msd = '.msd';
  msx = '.msx';

begin
  initindex;
  textcolor(14);

  (* enter name of file to be used from keybroad *)
  write('Please Enter Name of File ==> '); readln(name);

  (* create file name and file type *)
  newname  := name + dat;
  rname    := name + ixd;

  bname    := name + bom;
  bcname   := name + bcx;
  bmname   := name + bmx;
  bnname   := name + bnx;

  mname    := name + msd;
  msname   := name + msx;

  (* checking files procedure *)
  openfile(datf,newname,partregsize);
  if ok then
    begin
      clrscr;
      openindex(rfile,rname,6,0);

      openfile(bdatf,bname,partregsize);
      openindex(bcfile,bcname,12,0);
      openindex(bmfile,bmname,6,1);
      openindex(bnfile,bnname,6,1);

      openfile(mdatf,mname,partregsize);
      openindex(mfile,msname,6,0);
      closefile(datf);
      closeindex(rfile);

      closefile(bdatf);
      closeindex(bcfile);
```

```
      closeindex(bmfile);
      closeindex(bnfile);
      closefile(mdatf);
      closeindex(mfile);
    end;
    if not ok then
      begin      (* for creating new data/index files *)
        clrscr;
        gotoxy(17,12);
        write('Create a new file .. One more
               minute please');
        makefile(datf,newname,partregsize);
        makeindex(rfile,rname,6,0);

        makefile(bdatf,bname,partregsize);
        makeindex(bcfile,bcname,12,0);
        makeindex(bmfile,bmname,6,1);
        makeindex(bnfile,bnname,6,1);

        makefile(mdatf,mname,partregsize);
        makeindex(mfile,msname,6,0);

        closefile(datf);
        closeindex(rfile);

        closefile(bdatf);
        closeindex(bcfile);
        closeindex(bmfile);
        closeindex(bnfile);

        closefile(mdatf);
        closeindex(mfile);

      end;
end;



procedure dirlist;
(* This procedure is used to show directory of data/index
   files  *)
label 10;

type
  Char12arr             = array [ 1..12 ] of Char;
  String20              = string[ 20 ];
  RegRec =
    record
      AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags : Integer;
    end;

const
  dat = '.dat';
```

```
var
  Regs          : RegRec;
  DTA           : array [ 1..43 ] of Byte;
  Mask          : Char12arr;
  NamR          : String[20];
  Error, I      : Integer;
  namr1         : string[8];
  namr2         : string[12];
  X,an          : integer;

begin       (* main body of program DirList *)
  clrscr;
  x := 1;

  FillChar(DTA,SizeOf(DTA),0) (* Initialize the DTA
                                 buffer *)
  FillChar(Mask,SizeOf(Mask),0);  (* Initialize the mask *)
  FillChar(NamR,SizeOf(NamR),0);  (* Initialize the file
                                     name *)

  (* write directory screen  format *)
  WriteLn( ' MATERIAL REQUIREMENT PLANNING DIRECTORY
           LIST');
  WriteLn( ' ------------------------------------------
           -');
  Writeln;
  Regs.AX := $1A00;      (* Function used to set the DTA *)
  Regs.DS := Seg(DTA);   (* store the parameter segment
                            in DS *)
  Regs.DX := Ofs(DTA);   (*   "     "       "      offset
                            in DX *)
  MSDos(Regs);           (* Set DTA location *)
  Error := 0;
  Mask := '????????.???';    (* Use global search *)
  Regs.AX := $4E00;          (* Get first directory entry *)
  Regs.DS := Seg(Mask);      (* Point to the file Mask *)
  Regs.DX := Ofs(Mask);
  Regs.CX := 22;             (* Store the option *)
  MSDos(Regs);               (* Execute MSDos call *)
  Error := Regs.AX and $FF; (* Get Error return *)
  I := 1;          (* initialize 'I' to the first element *)
  if (Error = 0) then
    repeat
      NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
      I := I + 1;
    until not (NamR[I-1] in [' '..'~']) or (I>20);

    NamR[0] := Chr(I-1); (* set string length because
                            assigning *)

    (* by element does not set length *)
    while (Error = 0) do begin
      Error := 0;             (* directory entry *)
      Regs.CX := 22;          (* Set the file option *)
      MSDos( Regs );          (* Call MSDos *)
```

```
    Error := Regs.AX and $FF;  (* get the Error return *)
    I := 1;
    repeat
      NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
      I := I + 1;
    until not (NamR[I-1] in [' '..'~'] ) or (I > 20);
    NamR[0] := Chr(I-1);
    if (Error = 0) then
      begin
        textcolor(10);
        write(namr:14); textcolor(12); write(chr(124));
        if x = 5 then
          begin
            writeln;
            x := 1;
            goto 10;
          end
        else
          x := x +1;
      end;
10: end;
writeln; writeln; writeln;

checkfile;  (* call "CHECKFILE" procedure *)

end;
```

APPENDIX A-7

PROGRAM (DEBUGGING) CALL "SCREEN.MRP"

```
procedure screen_mrp;
(* This procedure is used to create MRP results screen *)

var
  a       : char;
  b,c,d,e : integer;

begin     (* create tabel *)
  clrscr;
  textcolor(2);
  gotoxy(1,1); write(char(218));
  for b := 2 to 80 do
    begin
      gotoxy(b,1); write(char(196));
    end;
  gotoxy(80,1); write(char(191));
  gotoxy(50,1); write(char(194));
  for b := 2 to 7 do
    begin
      gotoxy(1,b); write(char(179));
      gotoxy(50,b); write(char(179));
      gotoxy(25,b); write(char(179));
      gotoxy(80,b); write(char(179));
    end;
  gotoxy(1,7); write(char(192));
  for b := 2 to 80 do
    begin
      gotoxy(b,7); write(char(196));
    end;
  gotoxy(80,7); write(char(217));
  gotoxy(50,7); write(char(193));
  gotoxy(25,7); write(char(193));
  gotoxy(1,9); write(char(218));
  for b := 2 to 79 do
    begin
      gotoxy(b,9); write(char(196));
    end;
  gotoxy(80,9); write(char(191));
  gotoxy(24,9); write(char(194));
  c := 31;
  for b := 1 to 7 do
    begin
      gotoxy(c,9); write(char(194));
      c:= c + 7;
    end;
  c := 10;
  for b:= 1 to 14 do
    begin
      gotoxy(1,c); write(char(179));
      gotoxy(24,c); write(char(179));
      c := c + 1;
    end;
  c := 31;
```

```
for d := 1 to 8 do
  begin
    for e := 10 to 23 do
      begin
        gotoxy(c,e); write(char(179));
      end;
    c := c + 7;
  end;
c := 11;
for b := 1 to 7 do
  begin
    gotoxy(1,c); write(char(195));
    for d := 2 to 23 do
      begin
        gotoxy(d,c); write(char(196));
      end;
    gotoxy(24,c); write(char(197));
    for d := 25 to 79 do
      begin
        gotoxy(d,c); write(char(196));
      end;
    e := 31;
    for d := 1 to 7 do
      begin
        gotoxy(e,c); write(char(197));
        e := e + 7;
      end;
    gotoxy(80,c); write(char(180));
    c := c + 2;
  end;
  gotoxy(1,23); write(char(192));
  gotoxy(24,23); write(char(193));
  e := 31;
  for d := 1 to 7 do
    begin
      gotoxy(e,23); write(char(193));
      e := e + 7;
    end;
  gotoxy(80,23); write(char(217));
  textcolor(14);

  (* write informations in the table *)
  gotoxy(20,2); write('STOCK STATUS');
  gotoxy(60,2); write('MRP PLANNING');
  gotoxy(52,4); write('ITEM NUMBER : ');
  textcolor(3);
  gotoxy(2,3); write('Level Code    : ');
  gotoxy(2,4); write('Lot Size      : ');
  gotoxy(2,5); write('Lead Time     : ');
  gotoxy(2,6); write('On Hand       : ');
  gotoxy(26,3); write('Safety Stock  : ');
  gotoxy(26,4); write('Allocated     : ');
  gotoxy(26,5); write('On Order      : ');
  gotoxy(26,6); write('Due in Period : ');
  gotoxy(2,10); write('Period');
```

```
    gotoxy(25,10); write('  1');
    gotoxy(32,10); write('  2');
    gotoxy(39,10); write('  3');
    gotoxy(46,10); write('  4');
    gotoxy(53,10); write('  5');
    gotoxy(60,10); write('  6');
    gotoxy(67,10); write('  7');
    gotoxy(74,10); write('  8');
    gotoxy(2,12); write('Gross Requirements');
    gotoxy(2,14); write('Scheduled Receipts');
    gotoxy(2,16); write('Projected On Hand');
    gotoxy(2,18); write('Net Requirememts');
    gotoxy(2,20); write('Planned Order Receipts');
    gotoxy(2,22); write('Planned Order Releases');
    gotoxy(1,25);
end;
```

APPENDIX A-8

PROGRAM (DEBUGGING) CALL "SCREEN.S1"

```
procedure Frame(UpperLeftX, UpperLeftY, LowerRightX,
                LowerRightY: Integer);

(* This procedure is used to create frame for inventory
   record file, BOM relationship file and master
   production schedule file. (screen format) *)

var  I : Integer;

begin {Frame}
  GotoXY(UpperLeftX, UpperLeftY);
  Write(chr(218));
  for I := (UpperLeftX + 1) to (LowerRightX - 1) do
  begin
    Write(chr(196));
  end;
  Write(chr(191));
  for I := (UpperLeftY + 1) to (LowerRightY - 1) do
  begin
    GotoXY(UpperLeftX , I);  Write(chr(179));
    GotoXY(LowerRightX, I);  Write(chr(179));
  end;
  GotoXY(UpperLeftX, LowerRightY);
  Write(chr(192));
  for I := (UpperLeftX + 1) to (LowerRightX - 1) do
  begin
    Write(chr(196));
  end;
  Write(chr(217));
end;



procedure MakeScreen;
(* This procedure is use to create 3 windows for inventory
   record file screen, BOM relationship file screen and
   MPS file screen. *)

const
  Windows     = 3;
  Wtab        : array [1..Windows,1..5] of Integer
                { X0,Y0,X1,Y1,LineNo )
                = ((2,2,79,2,1),(2,6,55,20,1),
                   (58,6,79,20,1));
  Up          = true;
  Down        = false;

type
  MaxString   = string [255];

var
  I           : Integer;
  Ch          : char;
```

```
begin
  ClrScr;
  for i := 1 to windows do
    begin
      textcolor(11);

      (* call "FRAME" procedure *)
      Frame(Wtab[I,1] - 1, Wtab[I,2] - 1, Wtab[I,3] + 1,
      Wtab[I,4] + 1);
    end;
  end;
```

APPENDIX A-9

PROGRAM (DEBUGGING) CALL "SCREEN.S2"

```
procedure Frame1(UpperLeftX, UpperLeftY, LowerRightX,
          LowerRightY: Integer);
(* This procedure is used to create frame for all menu
   screens. *)

var
  I : Integer;

begin {Frame1}
  GotoXY(UpperLeftX, UpperLeftY);
  Write(chr(218));
  for I := (UpperLeftX + 1) to (LowerRightX - 1) do
  begin
    Write(chr(196));
  end;
  Write(chr(191));
  for I := (UpperLeftY + 1) to (LowerRightY - 1) do
  begin
    GotoXY(UpperLeftX , I);  Write(chr(179));
    GotoXY(LowerRightX, I);  Write(chr(179));
  end;
  GotoXY(UpperLeftX, LowerRightY);
  Write(chr(192));
  for I := (UpperLeftX + 1) to (LowerRightX - 1) do
  begin
    Write(chr(196));
  end;
  Write(chr(217));
end;



procedure M1Screen;
(* This procedure is used to create 1 window for all menu
   screens *)

const
  Windows     = 1;
  Wtab        : array [1..Windows,1..5] of Integer
              { X0,Y0,X1,Y1,LineNo }
              =  ((2,2,79,23,1));
  Up          = true;
  Down        = false;

type
  MaxString   = string [255];

var
  I           : Integer;
  Ch          : char;
```

```
begin
  ClrScr;
  for i := 1 to windows do
  begin
    textcolor(11);

    (* call "FRAME1" procedure *)
    Frame1(Wtab[I,1] - 1, Wtab[I,2] - 1, Wtab[I,3] + 1,
    Wtab[I,4] + 1);
  end;
end;
```

APPENDIX A-10

PROGRAM (DEBUGGING) CALL "MRP.PAS"

```
program mrp;

(* This is the main program of mrp program. The program
   prepares data structures for part list file, BOM file,
   master production schedule file, raw database file call
   " rdata1 ", assigned level code file, and final
   MRP result file. *)

type
  pl = record  (* Part list data structure *)
         partl          : string[6];
         status,
         lotsize,
         leadtime,
         onhand,
         safetystock,
         allocated,
         onorder,
         period         : integer;
       end;

  bl = record  (* BOM data structure *)
         pm,
         pn             : string[6];
         code           : string[12];
         status,
         qty            : integer;
       end;

  ms = record  (* Master production schedule data
                  structure *)
         partn    : string[6];
         status,
         p1,p2,p3,p4,p5,p6,p7,p8 : integer;
       end;

  rdata1 = record  (* raw data structure *)
             partname        : string[6];
             levelcode,
             allocate,
             safety_stock,
             on_hand,
             lead_time,
             lot_size,
             status,
             on_order,
             d_period        : integer;
           end;

  asign_code = record  (* assigned level code data
                          structure *)
                 partn    : string[6];
                 code,
                 status   : integer;
               end;
```

```
  mrp_planning = record  (* MRP results data structure *)
                   part_name    : string[6];
                   status,
                   g1,g2,g3,g4,g5,g6,g7,g8,
                   s1,s2,s3,s4,s5,s6,s7,s8,
                   h1,h2,h3,h4,h5,h6,h7,h8,
                   n1,n2,n3,n4,n5,n6,n7,n8,
                   or1,or2,or3,or4,or5,or6,or7,or8,
                   ol1,ol2,ol3,ol4,ol5,ol6,
                   ol7,ol8      : integer;
                 end;

  q = string[12];


CONST
  (* Identify size of B-Trees *)
  anw = '5';
  partregsize = 10000                  (* Max record size *)
  MAXDATARECSIZE     = partregsize;    (* Max record size *)
  MAXKEYLEN          = 15;             (* Max key size *)
  PAGESIZE           = 16;             (* Page size *)
  ORDER              =  8;             (* B-tree Order *)
  PAGESTACKSIZE      =  5;             (* Page buffer size *)
  MAXHEIGHT          =  5;             (* Max B-tree height *)

  recsize    = 128;
  bufsize    = 200;

  (* Calling Turbo Pascal subprograms, these
     subprograms are used to organize B-tree structure. *)

  (*$I ACCESS.BOX*)      (* used for data/index file setup
                           and data file maintenance *)
  (*$I GETKEY.BOX*)      (* used for search routines of
                           TURBO Access. *)
  (*$I ADDKEY.BOX*)      (* used for inserting keys into
                           index files *)
  (*$I DELKEY.BOX*)      (* used for deleting keys from
                           index files. *)

var
  p  : pl;
  b  : bl;
  m  : ms;
  as : assign_code;
  mp : mrp_planning;
  rl : rdata1;
  pn : string[6];
  a  : char;
  datf, bdatf, mdatf                  : datafile;
  rfile,bnfile,bmfile,bcfile, mfile   : indexfile;
```

```
   adatf, abdatf                          : datafile;
   arfile, abnfile,abmfile                : indexfile;

   wdatf,w1datf,w2datf                    : datafile;
   wfile,w1file,p1file,w2file             : indexfile;

   rd1datf                                : datafile;
   rx1file                                : indexfile;
   mrpdatf, pdatf,podatf,nmdatf           : datafile;
   mrpfile, pfile,pofile,nmfile           : indexfile;

   datref,
   buflen                                 : integer;
   name                                   : string[8];
   rname,newname,bname,bnname,bmname,bcname, mname,msname,
   aname,arname,abname,abnname,abmname,pname,pnname,nmname,
   nnmname   : string[12];

   (* Call subprograms to use with main program *)

   (*$I DIRLIST.U*)
   (*$I SCREEN.MRP*)
   (*$I SCREEN.S1*)
   (*$I DATA1.PL*)
   (*$I DATA2.BOM*)
   (*$I ASSIGN.LV*)
   (*$I SCREEN.S2*)
   (*$I DATA3.MS*)
   (*$I MRP.P*)



procedure database;
(* This procedure is used to create main MRP
   database menu. *)

const
  anw = '4';

var
  a : char;

begin
  repeat
    clrscr;
    textcolor(14);
    m1screen;
    textcolor(14);
    gotoxy(20,1);

    (* create screen format for MRP database main menu. *)
    writeln('   MATERIAL REQUIREMENT PLANNING DATA BASE');
```

```
    textcolor(10);
    gotoxy(25,8);  writeln('1.  Update Part List');
    gotoxy(25,10); writeln('2.  Update BOM Relationships');
    gotoxy(25,12);
    writeln('3.  Update Master Production Schedule');
    gotoxy(25,14); writeln('4.  Exit to Main Manual');
    textcolor(14);
    gotoxy(25,16);
    write('Enter the number ==> '); readln(a);

    case a of
      '1'  : begin              (* part list option *)
               makescreen;      (* call procedure
                                   "MAKESCREEN" *)
               update;          (* call procedure "UPDATE" *)
        end;
      '2'  : begin              (* BOM relationships option *)
               makescreen;      (* call procedure
                                   "MAKESCREEN" *)
               ebom;            (* call procedure "EBOM" *)
             end;
      '3'  : begin              (* MPS option *)
               makescreen;      (* call procedure
                                   "MAKESCREEN" *)
               master;           (* call procedure "MASTER" *)
             end;
    end;
  until anw = a;
end;


begin      (* M A I N   P R O G R A M *)
  repeat
    clrscr;
    mlscreen;
    textcolor(14);

    (* create screen MRP main menu *)
    gotoxy(20,1); writeln(' MATERIAL REQUIREMENT PLANNING
    MAIN MANUAL ');
    textcolor(10);
    textbackground(0);
    gotoxy(20,5);
    writeln('1. Create or Edit Data Base File');
    gotoxy(20,7);  writeln('2. MRP Data Base');
    gotoxy(20,9);  writeln('3. BOM Level Codes Procedure');
    gotoxy(20,11); writeln('4. MRP Procedure');
    gotoxy(20,13); writeln('5. Exit to DOS');
    textcolor(14);
    gotoxy(20,20);
    write('Enter the Number ==> '); readln(a);
```

```
    case a of
      '1' :  begin     (* create/edit database file
                          option *)
               dirlist;  (* call procedure "DIRLIST" *)
               database; (* call procedure "DATABASE" *)
             end;
      '2' : database; (* MRP database option, call procedure
                         "DATABASE" *)
      '3' : copy;     (* BOM level codes procedure option
                         and call procedure "COPY" *)
      '4' : computing_mrp;  (* MRP procedure option,
                               call procedure
                               "COMPUTING_MRP" *)
    end;
  until anw = a;
  clrscr;
end. (* E N D *)
```

APPENDIX B

THE PROGRAMS DESIGN LANGUAGE OR PSEUDOCODE

APPENDIX B-1

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

MRP.P

```
PROCEDURE NET_LFL

     This procedure is used to calculate net requirements,
     when lot size is defined as lot-for-lot.

     DECLARE  ng, nal, nsr, noh, nn, nh AS INTEGER

     BEGIN
       IF 0 = noh (projected on hand = 0) THEN
       BEGIN
         calculate net requirements (nn = ng+nal-nsr-noh)
         set projected on hands = 0
       END
       ELSE
       BEGIN
         IF noh > ng (projected on hand > gross
                      requirements) THEN
         BEGIN
           calculate projected on hands (nh = noh - ng)
           set net requirements = 0
         END
         ELSE
         BEGIN
           calculate net requirements (nn = ng - noh)
           set projected on hands = 0
         END
         ENDIF
       END
       ENDIF
     END
```

```
PROCEDURE NET_RQ

     This procedure is used to compute net requirements.

     DECLARE  nls, ng, nal, nsr, noh, nss, d, nn,
              nh, nor AS INTEGER

     BEGIN
       IF (d = 1) THEN
       BEGIN
         calculate projected on hands (nh = abs((ng + nal)
                                              - nsr - noh)
         set net requirements = 0
         set planned order receipts = net requirements
           (nor = nn)
       END
       ELSE
       BEGIN
         calculate projected on hand (nh = noh - ng)
         IF safety stock < projected on hand or projected
            on hand = 0  ((nss < nh) or (nh = 0)) THEN
         BEGIN
           calcualte projected on hand (abs(nh))
           set net requriements = 0
           set planned order receipts = net requirements
         END
         ELSE
         BEGIN
           calculate net requirements (nn = abs(nh - nss))
           set planned order receipts = lot size
             (nor = nls)
           calculate projected on hands (nh = nor + nh)
         END
         ENDIF
       END
       ENDIF
     END
```

```
PROCEDURE FIRST_LEVEL

     This procedure is used to perfrom the first level
     of MRP calculation.

     DECLARE  mr AS ARRAY
              g, nm, rp, n, h, orr, ol, nol,
              srr AS ARRAY
              bl, c, d, e, lc, al, ss, oh, lt, ls, sr,
              dp, cl, dl, control, q, sl, newoh AS INTEGER
              an AS CHAR
              pn, pneed, pmade AS CHAR STRING

     BEGIN
       set pointer to the top of file call "mrp.dat"
       FOR bl = 1 to number of records "mrp.inx" DO
         get next assigned code number for index file
           "mrp.inx"
         IF a = c (assigned code number) THEN
         BEGIN
           get all informations associated with code
             number from data file "mrp.dat"
           find part number in MPS index file
           IF ok THEN
           BEGIN
             get all informations associated with part
               number from MPS data file
             IF 1 = ls (lot size = 1) THEN
             BEGIN
               FOR d =  1 to 8 DO
                 call procedure net_lfl
                 set oh = projected on hands (h[d])
                 set orr[d] = net requirements (n[d])
                 set srr[d] = scheduled receipts (sr)
               ENDFOR
             END
             ELSE
             BEGIN
               FOR d = 1 to 8 DO
                 set srr[d] = scheduled receipts (sr)
                 call procedure net_rq
                 set oh = projected on hand (h[d])
                 set scheduled receipts (sr) = 0
               ENDFOR
             END
             ENDIF
             FOR d = 1 to 8 DO
               set planned order releases (ol[d]) = 0
             ENDFOR
             FOR d = 1 to 8 DO
               set e = d - lt
```

```
  IF e >= 1 THEN
  BEGIN
    set planned order releases (ol[e]) =
      planned order receipts (orr[d])
  END
  ELSE
  ENDIF
ENDFOR
write "please hit enter key for result"
read for answer (an)
IF lot size (ls) = 1 THEN
BEGIN
  write all results on format screen
  WITH mp DO
    perpare all MRP results for saving into
      MRP results data and index file (set
      all MRP results to database structures
      type call "mrp_planning"
    find part number in MRP results index
      file
    IF ok THEN
    BEGIN
      write error messages
    END
    ELSE
    BEGIN
      set mp.status = 0
      add new result associated with part
        number into MRP results data record
        file
      add part number into MRP results index
        file
      set pointer to the top of records in
        BOM relationship index file
      FOR c1 = 1 to number of records
                        (BOM data file) DO
        FOR d1 = 1 to 8 DO
          set planned order releases
            (nol[d1]) = ol[d1]
        ENDFOR
        get next part made from BOM index
          file
        IF part number (pn) = part made
                              (pmade) THEN
        BEGIN
          get part made fron BOM data file
          set part needed (pneed) = part made
                                    (b.pm)
          set q = b.qty
          find part need from part list index
            file
```

```
IF ok THEN
  BEGIN
    get information associated with
    part needed in part list data file
    find part needed from MPS index
      file
    IF ok THEN
      BEGIN
        get information associated
        with part needed from MPS
        data file
        FOR d1 = 1 to 8 DO
          calculate planned order
          releases (nol[d1] =(nol[d1]
          * q) + rp[d1] + nm[d1]
        ENDFOR
      END
    ELSE
      BEGIN
        FOR d1 = 1 to 8 DO
          calculate planned order
          releases (nol[d1] =(nol[d1]
          * q) + rp[d1]
        ENDFOR
      END
    ENDIF
  END
ELSE
  BEGIN
    find part needed from MSP index
      file
    IF ok THEN
      BEGIN
        get information associated
        with part neede from MPS data
        file
        FOR d1 = 1 to 8 DO
          calculate planned order
          releases (nol[d1] =(nol[d1]
          * q) + nm[d1]
        ENDFOR
      END
    ELSE
      BEGIN
        FOR d1 = 1 to 8 DO
          calculate planned order
          releases (nol[d1] =(nol[d1]
          * q)
        ENDFOR
      END
    ENDIF
  END
ENDIF
```

```
                find part needed from part list
                  index file
                IF ok THEN
                  BEGIN
                    delete part needed from part list
                         index file
                      delete part needed data file
                     END
                     ENDIF
                     set m.satatus = 0
                     set part number = part needed
                     add part number into part list data
                       file
                     add part number into part list
                       index file
                   ENDIF
                 END
               END
             END
             ELSE
             ENDIF
          END
        END
END
```

```
PROCEDURE COMPUTING_MRP

     This procedure is used to prepare back up copies
     data and index files before doing MRP calculation.

     DECLARE  100, 110 AS LABEL
              pdt, pnx, nmd, nmx AS CONST
              a, c, f1, f2, f3, f4, nc AS INTEGER
              dname AS CHAR STRING
              an AS CHAR

     BEGIN
       clear screen
       call procedure screen_mrp
       initindex
       make temporary data and index files call "cmrp.dat"
         and "cmrp.inx"
       set pname = name + pdt
       set pnname = name + pnx
       set nmname = name + nmd
       set nnmname = name + nmx
       call procedure filecopy (pname)
       call procedure filecopy (pnname)
       call procedure filecopy (nmname)
       call procedure filecopy (nnmname)
       set a = 1
       REPEAT
         open data and index files call "mrp.dat" and
           "mrp.inx"
         make data and index files call "pmrp.dat" and
           "pmrp.inx"
         open data and index files call "nmname" and
           "nnmname"
         open data and index files call "pname" and
           "pnname"
         call procedure first_level
         set a = a + 1
         set c = a
         set pointer to the top of index file "mrp.inx"
         FOR f1 = 1 to number of records in data file
                    "mrp.dat"
           set f3 = number of records in data file
                    "mrp.dat"
           get next assigned level code number from
             "mrp.inx"
           IF c = level code number (nc) THEN
           BEGIN
             close both data and index files call "pname"
               and "pnname"
             close both data and index files call "nmname"
               and "nnmname"
```

```
        close both data and index files call
          "pmrp.dat" and "pmrp.inx"
        close both data and index files call
          "mrp.dat" and "mrp.inx"
        call procedure filecopy ("pmrp.dat")
        call procedure filecopy ("pmrp.inx")
        go to LEVEL 100
      END
      ELSE
      BEGIN
        IF f1 = f3 THEN
        BEGIN
          close both data and index files call
            "mrp.dat" and "mrp.inx"
          close both MRP results data and index files
          close both data and index files call
            "pmrp.dat" and "pmrp.inx"
          close both data and index files call "pname"
          and "pnname"
          set c = 0
          go to LEVEL 100
        END
        ELSE
        BEGIN
          go to LEVEL 110
        END
        ENDIF
      END
  LEVEL 110 END
  LEVEL 100 UNTIL C = 0
  write "computing MRP completed.. hit enter key"
  read an (answer)
END
```

APPENDIX B-2

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

DATA1.PL

```
PROCEDURE ADDP

     This procedure is used to add part list, inventory
     record files, data and index files. The procedure
     calls subprograms from Turbo Tool Box
     call "ADDKEY.BOX".

     DECLARE  an AS CHAR
              a AS INTEGER

     BEGIN
       set textcolor (15)
       read part number
       find part number from part list index file
       IF ok THEN
       BEGIN
         create sound
         FOR a = 1 to 3
         DO
           write error messages
         ENDFOR
       END
     ELSE
       BEGIN
         read all information associated with part number
         add a new data record file
         add a new index record file
         write all information associated with part number
           on screen
       END
     ENDIF
   END
```

PROCEDURE DELETE

```
This procedure is used to delete part list, inventory
record files, data and index files. The procedure
calls subprogram from Turbo Tool Box call
"DELKEY.BOX".

DECLARE  a AS INTEGER

BEGIN
  set buffer length = 15
  read part number
  delete part number in part list index file
  IF ok THEN
    BEGIN
      delete part number data record file
      write record number after deleted
    END
  ELSE
    BEGIN
      create sound
      write error messages
    END
  ENDIF
END
```

PROCEDURE DEL

```
This procedure is used to clear unnecessary data on
screen.

BEGIN
  set textcolor(10)
  write "_______" at locations
END
```

```
PROCEDURE UPDATE

     This procedure is used to create screen format for
     inventory record files.  This procedure is a main
     option of inventory record files.

     DECLARE  anw AS CONST
              a, an, d, q AS CHAR
              a1 AS INTEGER
              data AS CHAR STRING

     BEGIN
       open part list data file
       open part list index file
       create screen format for inventory records file
       write command options
     REPEAT
       read command options
       CASE command options
         CASE 'a' BEGIN
                     call procedure del
                     call procedure addp
                  END
         CASE 'd' BEGIN
                     call procedure del
                     call procedure delete
                  END
         ENDCASE
     UNTIL anw = an
     close part list data and index files
     END
```

APPENDIX B-3

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

DATA2.BOM

PROCEDURE BCLEAR

This procedure is used to erase some parts of BOM relationship file screen format.

```
BEGIN
  write "_______" on screen at erase locations
END
```

PROCEDURE BADD

This procedure is used to add BOM relationship data and index files. The procedure calls subprograms from Turbo Tool Box call "ADDKEY.BOX".

```
DECLARE  bn, bnn, bbn, co AS CHAR STRING
         q, a1 AS INTEGER

BEGIN
  read part made number
  read part needed number
  create new code number by joining part made and part
  needed
  find code number from BOM relationship index file
  IF ok THEN
    BEGIN
      create sound
      FOR a1 = 1 to 3 DO
        write error messages
      ENDFOR
    END
  ELSE
    BEGIN
      read quality fo part needed
      add new BOM relationship data record file
      add BOM relationship code index record file
      add BOM relationship part made index record
        file
      add BOM relationship part needed index record
        file
      write all information on screen
      write number of records after added
    END
  ENDIF
  END
```

```
PROCEDURE BDELETE

     This procedure is used to delete BOM relationship data
     and index files.  The procedure calls subprograms from
     Turbo Tool Box call "DELKEY.BOX".

     DECLARE  bn, bm, co AS CHAR STRING
              a1 AS INTEGER

     BEGIN
       set buffer length = 15
       read part made number
       read part needed number
       create new code number by joining part made and part
       needed
       find and delete code number in BOM relationship
         index file
       IF ok THEN
         BEGIN
           delete part made index record file
           delate part needed index record file
           delate data record file
           write number of records after deleted
         END
       ELSE
         BEGIN
           create sound
           FOR a1 = 1 to 3 DO
             write error messages
           ENDFOR
         END
       ENDIF
       END
```

```
PROCEDURE EBOM

    This procedure is used to create BOM relationship file
    screen format.  This procedure is a main option of BOM
    relationship record files.

    DECLARE  anw AS CONST
             a, an, d, q AS CHAR

    BEGIN
      open BOM relationship data and all index files
      create BOM relationship record file screen format
      write command options
    REPEAT
      read an (command option)
      CASE an (command option)
        CASE 'a' BEGIN
                   call procedure bclear
                   call procedure badd
                 END
        CASE 'd' BEGIN
                   call procedure bclear
                   call procedure bdelete
                 END
      ENDCASE
    UNTIL anw = an (or answer = quit)
    close BOM relationship data and all index files
    END
```

APPENDIX B-4

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

ASSIGN.LV

PROCEDURE FILECOPY

This procedure is used to make back up copies database files.

```
DECLARE  bufsize, bufbytesize AS CONST
         source, dt AS FILE TYPE
         buffer AS ARRAY
         norec, remaining AS INTEGER

BEGIN
  assign file name to by copy (source file)
  reset source file
  assign new name copy file (back up file)
  create back up file in diskette
  set remaining = filesize of source file
  WHILE remaining > 0 DO
    IF bufsize <= remaining THEN
      BEGIN
        set norec = bufsize
      END
    ELSE
      BEGIN
        set norec = remaining
        read data from source file
        write data into back up file
        set remaining = remaining - norec
      END
    ENDIF
  ENDWHILE
  close back up database file
  END
```

PROCEDURE ASSIGNLV

This procedure is used to assigned level code for each part number. The procedure uses data from inventory record file and BOM relationship record file.

```
DECLARE  10, 20, 30 AS LABEL
         use4 AS CONST
         use1, use2, use3, u1, u2, a, d, c, e,
         count, count1, count2, x AS INTEGER
         data8, data10 AS CHAR STRING
         an AS CHAR

BEGIN
  clear screen
  write BOM LEVEL CODE PROCEDURE
  make temporary data and index files call "work1.dat"
  and "work1.inx"
  close both temporary data and index files
  set count = 0
LABEL 20 initindex
  open part list data and index files
  make temporary data and index files call "work.dat"
  and "work.inx"
  open temporary data and index files call "work1.dat"
  and "work1.inx"
  set use1 = number of records from part list file
  set pointer of part list at the top of records
  open BOM relationship data and all index files
  set use2 = number of records from BOM relationship
     file
  set pointer of BOM relationship at the top of
     records
  IF use4 = use2 THEN
    BEGIN
      write level code = count + 1
      set count1 = count1 + 1
      FOR e = 1 to use1 DO
        get next part number from part list file
        set data10 = data9
        set count2 = count1
        find data10 from temporary file call
          "work1.inx"
        IF ok THEN
          BEGIN
            write error messages
          END
        ELSE
          BEGIN
            set as.status = 0
            set as.partn = data10
            set as.code = count2
```

```
        add part number has been assigned into
           data and
        index file call "work1.dat" and
           "work1.inx"
        write part number has been assigned on
           screen
      END
    ENDIF
    close part list data and index files
    close BOM relationship data and index files
    close temporary data and index files call
       "work1.dat" and "work1.inx"
    go to LEVEL 30
  ENDFOR
ENDIF
set count = count + 1
write "LEVEL CODE" = count
set pointer of part list index file to the top
  of file
FOR a = 1 to use1 DO
  set pointer of BOM relationship part made
     index file to the top of file
  get next part number from part list index file
  set data3 = data1
  find data3 in BOM relationship part made
     index file
  IF ok THEN
    BEGIN
      set data10 = data1
      set count2 = count
      set p.part1 = data1
      find data1 in temporary index file call
        "work.inx"
      IF not ok THEN
        BEGIN
          set p.status = 0
          add part number has been assigned
             level code into temporary data and
             index files call "work.dat" and
             "work.inx"
          find data10 in temporary index file
             call "work1.inx"
          IF ok THEN
            BEGIN
              write error messages
            END
          ELSE
            BEGIN
              set as.status = 0
              set as.partn = data10
              set as.code = count2
```

```
                    add part number has been assigned
                       level code into data and index
                       files call "work1.dat"
                       and "work1.inx"
                    write part number has been assigned
                       level code on screen
                  END
                ENDIF
              END
            ENDIF
          END
        ENDIF
      ENDFOR
      set use3 = number of records "work.dat"
      set pointer of part number in "work.inx"
         at top of file
      FOR c = 1 to use3 DO
        get next part number in index file call
           "work.inx"
        set buflen = 15
        delete part number in part list data and
           index files
      ENDFOR
      close both part list data and index files
      FOR c = 1 to use3 DO
        get next part number in temporary index
           file call "work.inx"
        set data5 = data4
      LABEL 10 find data5 from BOM relationship
                  part needed index file
      IF ok THEN
        BEGIN
          get data from BOM relationship that
             associated with data5
          set data6 = b.pn
          set data7 = b.pm
          set data8 = b.code
          delete data6 from BOM relationship part
             needed index file
          delete data associated with data6 from BOM
             relationship data file
          delete data7 from BOM relationship part
             made index file
          go to LEVEL 10
        END
      ENDIF
    ENDFOR
    close all BOM relationship data and index files
    close both temporary data and index file call
       "work1.dat" and "work1.inx"
    go to LEVEL 20
    LEVEL 30 write assigned level code procedure
                completed
    read an
END
```

PROCEDURE CODEFILE

```
This procedure is used to perpare informations needed
before doing MRP calculation.  This procedure gets
all of data from both tomporary files call "work1.dat"
and part list file.  The procedure joins all data from
both files and make a new temporary data file call
"mrp.dat".

DECLARE  a1, a2, a3, z4 AS INTEGER
         z1, z2 AS CHAR STRING
         an AS CHAR

BEGIN
  open temporary data and index files call "work1.dat"
     and "work1.inx"
  open part list data and index files
  make temporary data and index files call "mrp.dat"
     and "mrp.inx"
  set a1 = number of records in "work1.dat" file
  set pointer to the top of file (work1.inx)
  FOR a2 = 1 to a1 DO
    get next part number from index file "work1.inx"
  IF ok THEN
    BEGIN
      get all informations associated with part number
         from "work1.dat"
      set z2 = as.partn
      set a3 = as.code
      set r1.partname = z2
      set r1.levelcode = a3
      find part number in part list file
      IF ok THEN
        BEGIN
          get all informations from part list file
          set all data to data structure type
          find part number in index "work2.inx" file
          IF not ok THEN
            BEGIN
              set r1.status = 1
              add part number into data file call
                 "mrp.dat"
              add level code number into index
                 file call "mrp.inx"
            END
          ELSE
            BEGIN
              write error messages
            END
          ENDIF
        END
      ELSE
```

```
        BEGIN
          write error messages
        END
      ENDIF
    END
  ENDIF
  close both temporary data and index files call
     "work1.dat" and "work1.inx"
  close both temporary data and index files call
     "mrp.dat" and "mrp.inx"
END
```

PROCEDURE COPY

This procedure is used to make some back up copy files such as inventory record file and BOM relationship file before perfroming assigned level code and perparing all data needed for MRP calculation.

```
DECLARE  dat, ixd, bom, bmx, bnx, adt, aix,
         aom, abm, abn AS CONST
         an, y AS INTEGER

BEGIN
  clear screen
  set newname = name + dat
  set rname = name + ixd
  set bname = name + bom
  set bmname = name + bmx
  set bnname = name + bnx
  set aname = name + adt
  set arname = name + aix
  set abname = name + aom
  set abmname = name + abm
  set abnname = name + abn
  call filecopy procedure, back up inventory
     record data and index files
  call filecopy procedure, back up BOM relationship
     data and all index files
  call assignlv procedure
  call filecopy procedure, back up inventory
     record data and index files
  call codefile procedure
END
```

APPENDIX B-5

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

DATA3.MS

```
PROCEDURE MDEL

     This procedure is used to erase some information
     on the master production schedule screen format file.

     BEGIN
       set textcolor (10)
       write "_______" at erase locations
     END
```

```
PROCEDURE MADDP

     This procedure is used to add master production
     shedule data and index file.  The procedure calls
     subprogram from Turbo Tool Box call "ADDKEY.BOX".

     DECLARE  an AS CHAR
              a AS INTEGER

     BEGIN
       read part number (pn)
       set m.partn = pn
       find part number in MPS index file
       IF ok THEN
         BEGIN
           create sound
           write error messages
         END
       ELSE
         BEGIN
           set p.status = 0
           read all informations associated with part
              number
           add all informations associated wiht part
              number into MPS data file
           add part number into MPS index file
           write number of records after added record
           write all informations on screen
         END
       ENDIF
     END
```

```
PROCEDURE MDELETE

     This procedure is used to delete MPS data and index
     records file. The procedure calls subprogram from
     Turbo Tool Box call "DELKEY.BOX".

     DECLARE  a AS INTEGER

     BEGIN
       set buffer length = 15
       read part number (pn)
       find and delete part number in MPS index
          record files
       IF ok THEN
         BEGIN
           delete data record associated with part number
             in MPS data record file
           write number of record after deleted
         END
       ELSE
         BEGIN
           create sound
           FOR a = 1 to 3 DO
             write error messages
           ENDFOR
         END
       ENDIF
     END
```

```
PROCEDURE MASTER

     This procedure is used to create screen format for
     master production schedule record file. This
     procedure is a main option of MPS record files.

     DECLARE  anw AS CONST
              an AS CHAR

     BEGIN
       open MPS data and index files
       write screen format of MPS record file
       REPEAT
         write command options
         read selected command option (an)
         CASE (an) command option
           CASE 'a' BEGIN
                      call procedure mdel
                      call procedure maddp
                    END
           CASE 'd' BEGIN
                      call procedure mdel
                      call procedure mdelete
                    END
       ENDCASE
      UNTIL anw = an (answer = quit)
      close MPS data and index record files
    END
```

APPENDIX B-6

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

DIRLIST.U

```
PROCEDURE CHECKFILE

     This procedure is used to check error data and
     index files, and also for creating new data and
     new index record files.

     DECLARE  dat, ixd, bom, bcx, bmx, bnx, msd,
              msx AS CONST

     BEGIN
       initindex
       set newname = name + dat
       set rname = name + ixd
       set bname = name + bom
       set bcname = name + bcx
       set bmname = name + bmx
       set bnname = name + bnx
       set mname = name + msd
       set msname = name + msx
       open inventory data record file
       IF ok THEN
       BEGIN
         clear screen
         open inventory index record file
         open BOM relationship data record file
         open all BOM relationship index files
         open MPS data record file
         open MPS index record file
         close both inventory data and index files
         close BOM relationship data and all index files
         close both MSP data and index files
       END
       ELSE
       BEGIN
         clear screen
         write "create a new file .. one more minute
                please"
         make both inventory data and index files
         make BOM relationship data and all index files
         make both MSP data and index files
         close both inventory data and index files
         close BOM relationship data and all index files
         close both MSP data and index files
       END
       ENDIF
     END
```

```
PROCEDURE DIRLIST

     This procedure is used to display a directory of all
     data and index files on the diskette.

     DECLARE  10 AS LABEL
              Char12arr AS ARRAY
              String20 AS CHAR STRING
              RegRec = RECORD
                         AX, BX, CX, DX, BP, SI, DI,
                         DS, ES, Flags AS INTEGER

              dat AS CONST

              Regs, Mask AS RECORD
              DTA AS ARRAY
              NamR, namr1, namr2 AS CHAR STRING
              Error, I, X, an AS INTEGER

     BEGIN
       clear screen
       set x = 1
       initialize the DTA buffer
       initialize the mask
       initialize the file name
       write "material requirement planning directory
             list"
       set Regs.AX = $1A00 (function used to set the DTA)
       set Regs.DS = Seg(DTA) (store the parameter segment
                               in DS)
       set Regs.DX = Ofs(DTA) (store the parameter offset
                               in DX)
       set MSDos = 0 (set DTA location)
       set Error = 0
       Mask = '????????.???' (use global search)
       set Regs.AX = $4E00 (get first directory entry)
       set Regs.DS = Seg(Mask) (point to the file mask)
       set Regs.DX = Ofs(Mask)
       set Regs.CX = 22 (store the option)
       set MSDos(Regs) (execute MSDos call)
       set Error = Regs.AX and $FF (get error return)
       set I = 1
       IF (Error = 0) THEN
       BEGIN
         REPEAT
           NamR[I] = Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]
           set I = I + 1
         UNTIL not (NamR[I-1] in [' '..'~']) or (I>20)
         set NamR[0] = Chr(I-1)
         WHILE (Error = 0) THEN
           set Error = 0 ( directory entry)
           set Regs.CX = 22 (set the file option)
```

```
      set MSDos(Regs) ( call MSDos)
      set Error = Regs.AX and $FF (get the error
                                   return)
      set I = 1
      REPEAT
        NamR[I] = Chr(Mem[Seg(DTA):OFs(DTA)+29+I])
        set I = I + 1
      UNTIL not (NamR[I-1] in [' '..'~']) or (I>20)
      set Namr[0] = Chr(I-1)
      IF (Error = 0) THEN
      BEGIN
        write (namr:14) and write chr(124)
        IF x = 5 THEN
        BEGIN
          set x = 1
          go to LABEL 10
        END
        ELSE
        BEGIN
          set x = x + 1
        END
        LEVEL 10 ENDIF
        call procedure checkfile
      END
      ENDIF
  END
  ENDIF
END
```

APPENDIX B-7

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

SCREEN.MRP

```
PROCEDURE SCREEN_MRP

     This procedure is used to create the screen mrp
     results.  The procedure uses charscter codes
     (ASII code) to draw the line.

     DECLARE  a, char(179), chr(191), char(192),
              char(193), char(194), char(195),
              char(196), char(197), char(217),
              char(218), AS STRING
              b, c, d, e, AS INTEGER

     BEGIN
       clear screen
       set textcolor(2)
       go to point x,y (1,1) on the screen
       FOR b = 2 to 80 DO
         go to x,y (b,1) write char(196)
       ENDFOR
       go to x,y (80,1) write char(191)
       go to x,y (50,1) write char(194)
       FOR b = 2 to 7 DO
         go to x,y (1,b) write char(179)
         go to x,y (50,b) write char(179)
         go to x,y (25,b) write char(179)
         go to x,y (80,b) write char(179)
       ENDFOR
       go to x,y (1,7) write char(192)
       FOR b = 2 to 80 DO
         go to x,y (b,7) write char(196)
       ENDFOR
       go to x,y (80,7) write char(217)
       go to x,y (50,7) write char(193)
       go to x,y (25,7) write char(193)
       go to x,y (1,9) write char(218)
       FOR b = 2 to 79 DO
         go to x,y (b,9) write char(196)
       ENDFOR
       go to x,y (80,9) write char(191)
       go to x,y (24,9) write char(194)
       set c = 31
       FOR b = 1 to 7 DO
         go to x,y (c,9) write char(194)
         set c = c + 7
       ENDFOR
       set c = 10
       FOR b = 1 to 14 DO
         go to x,y (1,c) write char(179)
         go to x,y (24,c) write char(179)
         set c = c + 1
       ENDFOR
       set c = 31
```

```
FOR d = 1 to 8 DO
  FOR e = 10 to 23 DO
    go to x,y (c,e) write char(179)
  ENDFOR
  set c = c + 7
ENDFOR
set c = 11
FOR b = 1 to 7 DO
  go to x,y (1,c) write char(195)
  FOR d = 2 to 23 DO
    go to x,y (d,c) write char(196)
  ENDFOR
  go to x,y (24,c) write char(197)
  FOR d = 25 to 79 DO
    go to x,y (d,c) write char(196)
  ENDFOR
  set e = 31
  FOR d = 1 to 7 DO
    go to x,y (e,c) write char(197)
    set e = e + 7
  ENDFOR
  go to x,y (80,c) write char(180)
  set c = c + 2
ENDFOR
go to x,y (1,23) write char(192)
go to x,y (24,23) write char(193)
set e = 31
FOR d = 1 to 7 DO
  go to x,y (e,23) write char(193)
  set e = e + 7
ENDFOR
go to x,y (80,23) write char(217)
set textcolor(14)
go to x,y (20,2) write "Storck Status"
go to x,y (60,2) write "MRP Planning"
go to x,y (52,4) write "Item Number"
set textcolor(3)
go to x,y (2,3) write "Level Code"
go to x,y (2,4) write "Lot Size"
go to x,y (2,5) write "Lead Time"
go to x,y (2,6) write "On Hand"
go to x,y (26,3) write "Safety Stock"
go to x,y (26,4) write "Allocated"
go to x,y (26,5) write "On Order"
go to x,y (26,6) write "Due in period"
go to x,y (2,10) write "Period"
go to x,y (25,10) write "1"
go to x,y (32,10) write "2"
go to x,y (39,10) write "3"
go to x,y (46,10) write "4"
go to x,y (53,10) write "5"
go to x,y (60,10) write "6"
go to x,y (67,10) write "7"
go to x,y (74,10) write "8"
```

```
        go to x,y (2,12) write "Gross Requirements"
        go to x,y (2,14) write "Scheduled Receipts"
        go to x,y (2,16) write "Projected On Hand"
        go to x,y (2,18) write "Net requirements"
        go to x,y (2,20) write "Planned Order Receipts"
        go to x,y (2,22) write "Planned Order Releases"
        go to 1,25)
END
```

APPENDIX B-8

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

SCREEN.S1

```
PROCEDURE FRAME

     This procedure is a subprogram of the procedure
     MakeScreen, the procedure is used ACII code
     to draw the lines.


     DECLARE  UpperLeftX, UpperLeftY, LowerRightX,
              LowerRightY, I AS INTEGER

     BEGIN
       go to x,y (UpperLeftX,UpperLeftY); write char(218)
       FOR I = (UpperLeftY + 1) to (LowerRightX - 1) DO
         write char(196)
       ENDFOR
       write char(191)
       FOR I = (UpperLeftY + 1) to (LowerRightX - 1) DO
         go to x,y (UpperLeftX, I) write char(179)
         go to x,y (LowerRightX, I) write char(179)
       ENDFOR
       go to x,y (UpperLeftX, LowerRightY) write char(192)
       FOR I = (UpperLeftX + 1) to (LowerRightX - 1) DO
         write char(196)
       ENDFOR
       write char(217)
     END
```

PROCEDURE MAKESCREEN

This procedure is used to create a frame on the screen to used with prat list, BOM relationships, and master production schedule database entry. The procedure calls subprogram Frame.

```
DECLARE  Windows AS CONST
         Wtab AS ARRAY
         Up, Down AS BOOLEAN
         MaxString, Ch AS CHAR
         I AS INTEGER

BEGIN
  clear screen
  FOR i = 1 to windows DO
    set textcolor(11)
    call subprogram frame
  ENDFOR
END
```

APPENDIX B-9

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

SCREEN.S2

```
PROCEDURE FRAME1

     This procedure is a subprogram of the procedure
     M1Screen, the procedure is used ACII code to draw
     the lines of material requirement planning main
     manual screen.

     DECLARE  UpperLeftX, UpperLeftY, LowerRightX,
              LowerRightY, I AS INTEGER

     BEGIN
       go to x,y (UpperLeftX,UpperLeftY); write char(218)
       FOR I = (UpperLeftY + 1) to (LowerRightX - 1) DO
         write char(196)
       ENDFOR
       write char(191)
       FOR I = (UpperLeftY + 1) to (LowerRightX - 1) DO
         go to x,y (UpperLeftX, I) write char(179)
         go to x,y (LowerRightX, I) write char(179)
       ENDFOR
       go to x,y (UpperLeftX, LowerRightY) write char(192)
       FOR I = (UpperLeftX + 1) to (LowerRightX - 1) DO
         write char(196)
       ENDFOR
         write char(217)
     END
```

PROCEDURE M1SCREEN

This procedure is used to create a frame on the material requirements planning main menu screen. The procedure calls subprogram frame1 to draw the lines.

```
DECLARE  Windows AS CONST
         Wtab AS ARRAY
         Up, Down AS BOOLEAN
         MaxString, Ch AS CHAR
         I AS INTEGER

BEGIN
  clear screen
  FOR i = 1 to windows DO
    set textcolor(11)
    call subprogram frame1
  ENDFOR
END
```

APPENDIX B-10

PROGRAM DESIGN LANGUAGE OR PSEUDOCODE

MRP.PAS

```
PROGRAM MRP

    This is the main program of MRP program.  The program
    prepares data structures for inventory record file,
    BOM relationship record file, master production
    schedule record file, rew database record file call
    "rdata1", assigned level code record file, and final
    MRP results record file.

    DECLARE
      p1 = RECORD
             part1 AS CHAR STRING
             status, lotsize, leadtime, onhand,
             safetystock, allocated, onorder,
             period AS INTEGER
      b1 = RECORD
             pm, pn, code AS CHAR STRING
             status, qty AS INTEGER
      ms = RECORD
             partn AS CHAR STRING
             status, p1, p2, p3, p4, p5, p6, p7,
             p8 AS INTEGER
      rdata1 = RECORD
                 partname AS CHAR STRING
                 levelcode, allocate, safety_stock,
                 on_hand, lead_time, lot_size, status,
                 on_order, d_period AS INTEGER
      asign_code = RECORD
                     partn AS CHAR STRING
                     code, status AS INTEGER
      mrp_planning = RECORD
                       part_name AS CHAR STRING
                       status, g1, g2, g3, g4, g5, g6,
                       g7, g8, s1, s2, s3, s4, s5, s6,
                       s7, s8, h1. h2, h3, h4, h5, h6,
                       h7, h8, n1, n2, n3, n4, n5, n6,
                       n7, n8, or1, or2, or3, or4, or5,
                       or6, or7, or8, o11, o12, o13, o14,
                       o15, o16, o17, o18 AS INTEGER
      q AS CHAR STRING
    anw, partregsize, maxdatarecsize, maxkeylen,
    pagesize, order, pagestacksize, maxheight, recsize,
    bufsize AS CONST

    call subprogram "ACCESS.BOX"
    call subprogram "GETKEY.BOX"
    call subprogram "ADDKEY.BOX"
    call subprogram "DELKEY.BOX"

    p, b, m, as, mp, r1 AS RECORD TYPE
    pn AS CHAR STRING
```

```
datf, bdatf, mdatf, adatf, abdatf, wdatf, w1datf,
w2datf, rd1datf, mrpdatf, pdatf, podatf,
nmdatf AS DATAFILE

rfile, bnfile, bmfile, bcfile, mfile, arfile,
abnfile, abmfile, wfile, w1file, w2file, rxfile,
mrpfile, pfile, pofile, nmfile AS INDEXFILE

datref, buflen AS INTEGER

name, rname, newname, bname, bnname, bmname, bcname,
mname, msname, aname, arname, abname, abnname,
abmname, pname, pnname, nmname,
nnmname AS CHAR STRING

call subprogram "DIRLIST.U"
call subprogram "SCREEN.MRP"
call subprogram "SCREEN.S1"
call subprogram "DATA1.PL"
call subprogram "DATA2.BOM"
call subprogram "ASSIGN.LV"
call subprogram "SCREEN.S2"
call subprogram "DATA3.MS"
call subprogram "MRP.P"

BEGIN
  REPEAT
    clear screen
    call procedure mlscreen
    gotoxy(20,1) write "material requirement planning
                        main manual"
    gotoxy(20,5) write "1. create or edit data base
                        file"
    gotoxy(20,7) write "2. MRP data base"
    gotoxy(20,9) write "3. BOM level codes procedure"
    gotoxy(20,11) write "4. MRP procedure"
    gotoxy(20,13) write "5. exit to DOS"
    gotoxy(20,20) write "enter the number ==>"
    read option number (a)
    CASE selected option number (a)
      CASE '1' BEGIN
                 call procedure dirlist
                 call procedure database
               END
      CASE '2' BEGIN
                 call procedure database
               END
      CASE '3' BEGIN
                 call procedure copy
               END
      CASE '4' BEGIN
                 call computing_mrp
               END
```

```
    ENDCASE
  UNTIL anw = a (answer = 5)
END
```

```
PROCEDURE DATABASE

    This procedure is used to create main MRP database
    menu.  And create option selected commands.

    DECLARE  anw AS CONST
             a AS CHAR

    BEGIN
      REPEAT
        clear screen
        call procedure m1screen
        write "material requirement planning data base"
        gotoxy(25,8) write "1.  update part list"
        gotoxy(25,10) write "2. update BOM relationships
        gotoxy(25,12) write "3. update master
                             production schedule"
        gotoxy(25,14) write "4. exit to main manual"
        gotoxy(25,16) write "enter the number ==>"
        read selected option command (a)
        CASE selected option command (a)
          CASE '1' BEGIN
                     call program makescreen
                     call program update
                   END
          CASE '2' BEGIN
                     call program makescreen
                     call ebom
                   END
          CASE '3' BEGIN
                     call program makescreen
                     call program master
                   END
        ENDCASE
      UNTIL anw = a (answer = 4)
    END
```

REFERNCES

1. Orlicky, Joseph, Material Requirements Planning, 1975, p. 3.
2. Orlicky, Joseph, Material Requirements Planning, 1975, pp. 35-36.
3. APICS Training Aid, Material Requirements Planning, p. 1.
4. APICS Training Aid, Material Requirements Planning, p. 1.
5. APICS Training Aid, Material Requirements Planning, p. 1.
6. Plossl, George W. "MRP Yesterday, Today and Tomorrow", Production and Inventory Management, Third Qtr, 1980, p. 1.
7. Plossl, George W. "MRP Yesterday, Today and Tomorrow," Production and Inventory Management, Third Qtr, 1980, p. 1.
8. Oliver W. Wight, "Material Requirements Planning", Production and Inventory Management in the Computer Age, 1974, p. 31.
9. Karen Marie May, MRP Explosion for the Commodore Computer, 1983, p. 8.
10. Ned W. Brenizer, "The Bottom Line of MRP Begins at the Top", 20th Annual Conference Proceedings, November 2-4, 1977, p. 271.

11. Plossl, George W. "MRP Yesterday, Today and Tomorrow", Production and Inventory Management, Third Qtr, 1980, p. 2.

12. Plossl, George W. "MRP Yesterday, Today and Tomorrow", Production and Inventory Management, Third Qtr, 1980, p. 2.

13. Plossl, George W. "MRP Yesterday, Today and Tomorrow", Production and Inventory Management, Third Qtr, 1980, p. 2.

14. Plossl, George W. "MRP Yesterday, Today and Tomorrow", Production and Inventory Management, Third Qtr, 1980, p. 2.

15. Plossl, George W. "MRP Yesterday, Today and Tomorrow", Production and Inventory Management, Third Qtr, 1980, p. 2.

16. Joseph A. Orlicky and William B. McDonald IBM Corporation, "Net Change MRP Implementation Considerations", 20th Annual Conference Proceedings, November 2-4, 1977, p. 294.

17. Joseph A. Orlicky and William B. McDonald IBM Corporation, "Net Change MRP Implementation Considerations", 20th Annual Conference Proceedings, November 2-4, 1977, p. 294.

18. Joseph A. Orlicky and William B. McDonald IBM Corporation, "Net Change MRP Implementation Considerations", 20th Annual Conference Proceedings, November 2-4, 1977, p. 294.

19. Joseph A. Orlicky and William B. McDonald IBM Corporation, "Net Change MRP Implementation Considerations", 20th Annual Conference Proceedings, November 2-4, 1977, p. 294.

20. Karen Marie May, MRP Explosion for the Commodore Computer, 1983, p. 2.

21. Roger S. Pressman, "Program Design Language", Software Engineering: A Practitioner's Approach, pp. 253-261.

22. APICS Training Aid, "Net Requirements and Planned Orders",Material Requirements Planning, p. 44.

23. APICS Training Aid, "Net Requirements and Planned Orders",Material Requirements Planning, p. 44.

MRP EXPLOSION FOR THE IBM COMPUTER

by

SUNSERN CHUANPRAPUN

B.S., King Mongkut's Institute of Technology, 1981

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

ABSTRACT

With the microcomputer becoming ever more popular, with more memory to solve problems. A study was needed to determine the practicality and feasibility of putting an MRP program on the microcomputer.

The development of MRP software for the IBM or IBM compatible under PC-DOS or MS-DOS operating system is used to perform the basic MRP explosion. The program are written in the programming language known as Turbo Pascal. B-tree technique are used to create and organize all of the databases.

The MRP program uses input from part list records file, bill of material records file, and assigned level code technique introduced by L. E. Grosh to assigned level code for each part number. The MRP explosion uses input from assigned level code, data from part list, bill of material, master production schedule records file, and the MRP explosion technique introduced by APICS training aid to generate the gross requirements, the scheduled receipts, the projected on hand, the net requirements, the planned order receipts, and the planned order releases for each part number.

It was found from this study that RAMdisk could increased the speed of the MRP program. However, the 2 Mgbyte RAMdisk does not have enough memory to solve large problems. To solve a large problem a hard disk drive is recommended. However the results of using a hard disk (access time 65Ms) was the decreased speed of the MRP program.